

Activating formal verification of deep reinforcement learning policies by model checking bisimilar latent space models

Delgrange, Florent

Publication date:
2024

License:
CC BY-SA

Document Version:
Final published version

[Link to publication](#)

Citation for published version (APA):
Delgrange, F. (2024). *Activating formal verification of deep reinforcement learning policies by model checking bisimilar latent space models*. [PhD Thesis, Vrije Universiteit Brussel, Universiteit Antwerpen]. VUB Press.

Copyright

No part of this publication may be reproduced or transmitted in any form, without the prior written permission of the author(s) or other rights holders to whom publication rights have been transferred, unless permitted by a license attached to the publication (a Creative Commons license or other), or unless exceptions to copyright law apply.

Take down policy

If you believe that this document infringes your copyright or other rights, please contact openaccess@vub.be, with details of the nature of the infringement. We will investigate the claim and if justified, we will take the appropriate steps.



Faculty of Science and Bio-Engineering Sciences
Department of Computer Science
Artificial Intelligence Laboratory



Faculty of Science
Department of Computer Science

Activating Formal Verification of Deep Reinforcement Learning Policies by Model Checking Bisimilar Latent Space Models

Dissertation submitted in fulfillment of the requirements for the degree of Doctor of Science: Computer science

Florent Delgrange

Brussels, June 2024

Promotors: Prof. Dr. Ann Nowé (Vrije Universiteit Brussel)
Prof. Dr. Guillermo A. Pérez (Universiteit Antwerpen)

© Florent Delgrange

2024 Uitgeverij VUBPRESS Brussels University Press

VUBPRESS is an imprint of Owl Press (an academic imprint of Borgerhoff & Lamberigts nv)

Keizerslaan 34

B-1000 Brussels

Tel. +32 (0)2 289 26 50

E-mail: info@owlpress.be

www.owlpress.be

ISBN 9789493387683

NUR 992

Legal deposit D2024/11.089/270

All rights reserved. No parts of this book may be reproduced or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written permission of the author.

Members of the Jury

Prof. Dr. Bernard Manderick	Vrije Universiteit Brussel, BE (Chair)
Dr. Roxana Rădulescu	Vrije Universiteit Brussel, BE Utrecht University, NL (Secretary)
Prof. Dr. Ann Nowé	Vrije Universiteit Brussel, BE (Promotor)
Prof. Dr. Guillermo A. Pérez	Universiteit Antwerpen, BE (Promotor)
Prof. Dr. Bart Goethals	Universiteit Antwerpen, BE
Prof. Dr. Nils Jansen	Radboud University Nijmegen, NL Ruhr-Universität Bochum, DE
Prof. Dr. Pablo Samuel Castro	Google Université de Montréal, CA

Summary

Intelligent agents are computational entities that autonomously interact with an environment to achieve their design objectives. On the one hand, reinforcement learning (RL) encompasses machine learning techniques that allow agents to learn by trial and error a control policy, prescribing how to behave in the environment. Although RL is proven to converge to an optimal policy under some assumptions, the guarantees vanish with the introduction of advanced techniques, such as deep RL, to deal with high-dimensional state and action spaces. This prevents them from being widely adopted in real-world safety-critical scenarios.

On the other hand, formal methods are mathematical techniques that provide guarantees about the correctness of systems. In particular, model checking allows formally verifying the agent’s behaviors in the environment. However, this typically relies on a formal description of the interaction, as well as conducting an exhaustive exploration of the state space. This poses significant challenges because the environment is seldom explicitly accessible. Even when it is, model checking suffers from the curse of dimensionality and struggles to scale to high-dimensional state and action spaces, which are common in deep RL.

In this thesis, we leverage the strengths of deep RL to handle realistic scenarios while integrating formal methods to provide guarantees on the agent’s behaviors. Specifically, we activate formal verification of deep RL policies by learning a latent model of the environment, over which we distill the deep RL policy. The outcome is amenable for model checking and is endowed with bisimulation guarantees, which allows to lift the verification results to the original environment.

Beyond distillation, we show that our method is also useful for learning representation in the context of deep RL, facilitating the learning of the policy in complex environments. In particular, we present a framework for partially observable environments.

We finally show how our method can be leveraged in the context of synthesis, i.e., the automatic generation of controllers from logical specifications with formal guarantees. Precisely, we present how deep RL components learned via our latent space models facilitate synthesis in typically intractable environments.

Samenvatting

Intelligente agenten zijn computationele entiteiten die autonoom interageren met een omgeving om hun doelstellingen te bereiken. Reinforcement learning (RL) verwijst naar technieken voor machinaal leren die agenten in staat stellen om een policy te leren op basis van een gegeven doelstelling. Hoewel bewezen is dat RL convergeert naar een optimale policy onder bepaalde aannames, verdwijnen de garanties met de introductie van geavanceerde technieken, zoals deep RL, welke nodig is om met hoog-dimensionale toestands- en actieruimtes om te gaan. Dit voorkomt dat ze op grote schaal worden toegepast in veiligheidskritische scenario's.

Formele methoden wiskundige technieken die garanties geven over de correctheid van systemen. Met name “model checking” maakt het mogelijk om het gedrag van de agent in de omgeving formeel te verifiëren. Dit is echter meestal afhankelijk van een formele beschrijving van de interactie, evenals het uitvoeren van een volledige exploratie van de toestandsruimte. Dit brengt aanzienlijke uitdagingen met zich mee, omdat de omgeving zelden expliciet toegankelijk is. Zelfs als dat wel het geval is, heeft “model checking” te lijden onder de vloek van de dimensionaliteit en heeft het moeite om op te schalen naar hoog-dimensionale toestands- en actieruimten, die vaak voorkomen in diepe RL.

In dit proefschrift maken we gebruik van de sterke punten van deep RL om realistische scenario's aan te kunnen, terwijl we formele methoden integreren om garanties te bieden voor het gedrag van de agent. Specifiek faciliteren we formele verificatie van de deep RL policy door het leren van een latent model van de omgeving alsook de deep RL policy. De “model checking” wordt dan in het gedistilleerde latente model wordt uitgevoerd. Aan gezien het gedistilleerde latente model voorzien is van bisimulatiegaranties, zijn de garanties overdraagbaar naar de oorspronkelijke omgeving en policy.

Naast destillatie laten we zien dat onze methode ook nuttig is voor het leren van representaties in de context van diepe RL, waardoor het leren van de policy in complexe omgevingen wordt vergemakkelijkt. In het bijzonder presenteren we een raamwerk voor partieel observeerbare omgevingen.

Tot slot laten we zien hoe onze methode kan worden gebruikt in de context van synthese, d.w.z. het automatisch genereren van controllers uit logische specificaties met

formele garanties. We laten met name zien hoe deep RL-componenten, geleerd via onze latente modellen, synthese vergemakkelijken in omgevingen die doorgaans moeilijk te analyseren zijn.

Acknowledgments

This dissertation results from a long and challenging journey, and I am grateful to many people who have supported me along the way.

First of all, I would like to thank my advisors, Ann Nowé and Guillermo A. Pérez. Ann, I am deeply grateful to you for allowing me to work with you and to join the AI Lab, for your advice, your support and the trust you have placed in me. Guillermo, thank you very much for your support, your wise advice, the time you spent reading and discussing my work, and the valuable comments you provided. I would also like to thank you both for the freedom of research you gave me. When I presented my projects to you, you were always enthusiastic about my ideas, allowed me to carry them out and helped me to realize their potential. This has certainly allowed me to grow scientifically in a very positive and supportive environment.

I would also like to thank the members of my doctoral examination committee, Bernard Manderick, Roxana Rădulescu, Bart Goethals, Nils Jansen, and Pablo Samuel Castro, who agreed to take the time to review this thick pile of material and gave me very positive feedback which helped me to improve the quality of this dissertation.

I never thought I would enjoy research as much as I did during my PhD and this is most certainly due to the great people I had the pleasure to work with. The AI Lab is a fantastic place to work and I am grateful to all the members of the lab for doing their part in creating a stimulating and friendly environment. This allowed us to incubate many ideas and develop them into successful projects — and some less successful ones, but that's part of the game (a special tribute to our Francky project, Mathieu). I would particularly like to thank Youri, Pieter, and Mathieu who welcomed me in the lab and made me feel quickly at home. My journey would not have been the same without my teammates and friends. Mathieu (already the third time I mention you), thanks for the discussions, projects, and laughs we shared before you moved to the other side of the planet. Raphael, thanks for the great collaboration, the many insightful discussions, and, most importantly, the fun we had. I'm glad to have started this journey with you and I'm happy to share great works. Willem, thanks for your enthusiasm, for organizing so many great research events with Raphael, and for lifting the mood in the office with your

Acknowledgments

dynamism, which gets the day off to a flying start. To quote your most pertinent line, *tu mets la patate*. Roxana, I appreciated the Friday mornings you spent with us discussing research and the many meaningful comments you provided on our work. Hicham and Andrea, thanks for the in-depth weekly discussions we've had on some high-velocity sports topics.

I also thank all my co-authors. Anna, Christian, and Guy, thank you for sharing so many interesting discussions, which led to great work and helped me rethink, question, and refine some of the frameworks presented in this dissertation.

I am also lucky to have great friends with whom I've grown up and shared some exceptional moments since my studies at the university. Amandine, Clément, Clémentine, Lorentz, Marie, Martin, and Nicolas, thanks for all the great times we've had together, which allowed me to switch off when I needed to.

Undeniably, I am thankful to my parents who raised me into who I am and gave me the necessary education to reach this milestone. I also thank all my family for their support.

Finally, thank you Carolane for believing so much in me, for your love, and for all the formidable adventures we share.

Contents

Members of the Jury	3
Summary	5
Samenvatting	7
Acknowledgments	9
Contents	11
1 Introduction	15
2 Background	27
1 Decision Making under Uncertainty	27
1.1 A Brief Tour through Probability Spaces	28
1.2 Markov Models	31
1.3 Policies	34
1.4 Limiting Behaviors	40
1.5 Return and Value Functions	46
1.6 Solving MDPs for the Expected Return	48
2 Reinforcement Learning	54
2.1 Episodic Processes	56
2.2 Value-based Methods	64
2.3 Actor-critic Methods	70
2.4 Are Deep RL Algorithms Reliable?	77
3 Model Checking and Synthesis	78
3.1 Linear Temporal Logic	79
3.2 Reachability	82

3.3	Stochastic Model Checking of LTL Properties	86
3.4	Bad News: LTL is not Learnable	97
3.5	Valued λ -calculus	101
3.6	Linear Properties in VLC	105
3.7	VLC's Linear Properties are PAC Learnable	110
3.8	Model Checking ω -regular Properties through VLC	111
4	Bisimulation	117
4.1	The Bisimulation Relation	118
4.2	Probability Metrics	124
4.3	Bisimulation Pseudometrics	128
5	Concluding Discussion	131
3	Distillation of RL Policies with Formal Guarantees	133
1	Latent Space Modeling	136
1.1	Local Losses	139
1.2	Bisimulation Bounds	140
1.3	Quantifying the Quality of the Abstraction	153
2	Variational MDP	161
2.1	The Evidence Lower Bound	163
2.2	The Variational Architecture	164
2.3	Back to the Local Setting	166
2.4	Variational Distributions	169
2.5	Posterior Collapse	173
2.6	Experiments	176
3	Wasserstein Auto-encoded MDP	180
3.1	The Discrepancy Measure	181
3.2	The WAE-MDP Objective	183
3.3	Optimizing the Regularizers	189
3.4	Discrete Latent Model	191
3.5	The Latent Metric	197
3.6	Experiments	200
4	Concluding Discussion	203
4	Learning to Believe in POMDPs via Reliable Latent Models	207
1	Decision Making with Imperfect Observations	213
2	Learning the Dynamics	215
2.1	The Latent POMDP Dynamics	216

2.2	Stationarity over Histories	219
2.3	Local Losses	225
2.4	Value Difference Bounds	227
3	Learning to Believe	232
4	Experiments	236
5	Concluding Discussion	239
5	Synthesis from Deep RL Components	241
1	Problem Setting	244
1.1	Problem Formulation	245
2	Obtaining Latent Low-Level Policies using DRL	249
2.1	Latent Space Models and Guarantees	249
2.2	Obtaining Latent Policies During Training	251
3	Obtaining a High-Level Planner	252
3.1	Planner Synthesis	252
3.2	Reusable Latent Policies	263
4	Experimental Evaluation	273
5	Concluding Discussion	276
6	Conclusion	279
	Bibliography	285
	Author’s Publications	317
	Appendix	319
A	Details on VAE-MDPs Experiments	319
A.1	Details on Policy Distillation	319
A.2	Setup	320
A.3	Hyperparameters	320
A.4	Labeling functions	321
A.5	Reproducibility of the Results	321
B	Details on WAE-MDPs Experiments	321
B.1	Setup	321
B.2	Hyperparameters	321
B.3	Reproducibility of the Results	324
C	The Wasserstein Believer: Supplementary Material	327
C.1	Proof of Theorem 30	327

CONTENTS

C.2	Hyperparameters	336
D	Synthesis from DRL components	338
D.1	Details on the Experiments	338
Index		343

Introduction

Intelligent agents are computational entities that autonomously interact with an *environment*. The environment often refers to the physical world, a complex simulation of the latter, a virtual one, or even a software system. Agents are usually designed to achieve specific objectives within their environment, such as controlling a robot [Xiao et al., 2022; Haarnoja et al., 2024], playing video games [Mnih et al., 2015; Wurman et al., 2022], managing a power grid [Mason and Grijalva, 2019; Ceusters et al., 2021], or even mitigating epidemics [Libin et al., 2020; Reymond et al., 2024] and controlling nuclear fusion [Degrave et al., 2022].

Suppose you, dear reader, want to design a robot that navigates to a specific location without colliding with obstacles. One way to achieve this is to program the robot with a set of rules that dictate how to move based on the sensor readings. If you consider a toy environment that consists of the map in which the agent operates divided into a grid (a “grid world”), the number of possible states (number of cells) and actions (the 4 directions) may be small, and the designer can manually specify those rules according to the position of the agent in the grid. However, this approach is hardly tractable, if not impossible in realistic scenarios. Indeed, the environment can be

- *complex*: it may consist of your office, house, garden, or even the park down your street;
- *dynamic*: people may enter and leave, change room configuration;

- *uncertain*: this could relate to the unexpected presence of humans or obstacles in the room, doors being locked, the accuracy of the sensors; or
- *large, continuous* and embed *high-dimensional spaces*: the agent may need to process images or videos and control its joint actuators.

Reinforcement learning. A suitable alternative is to design an agent that learns how to navigate by itself, by continuously interacting with the environment. This is the core idea behind *reinforcement learning* (RL, e.g., Sutton and Barto, 2018; Szepesvári, 2010): the agent learns a *control policy*, by trial and error, to maximize some numerical *reward signal* expressing a goal or a desirable behavior (Figure 1.1). Then, your role is not to explicitly program how the robot should behave but to design the way the robot learns to navigate (i.e., the *learning algorithm*) and the reward signal that guides the learning process.

A simple analogy to understand RL is the way pets are trained. When you teach a dog to sit, you give it a treat when it sits down. Then, the dog explores different behaviors until it learns that sitting down is associated with a reward, and it will repeat this behavior to get the treat.

Guarantees. Since your goal is to design a robot that navigates safely, you may want to ensure that the robot does not fall, does not collide with obstacles, and eventually reaches the target location. In simple environments (precisely, those that can be modeled through tabular data structures), RL algorithms are guaranteed to converge to an optimal policy under some assumptions [Tsitsiklis, 1994]. However, when considering more complex scenarios, *neural networks* are often used to process high-dimensional inputs (e.g., images or solutions of differential equations corresponding to states of a dynamical system) and outputs (e.g., continuous control signals). In that case, all the convergence guarantees vanish. The introduction of neural networks in RL is known as *deep reinforcement learning* (deep RL, e.g., Mnih et al., 2015) as they allow to automatically

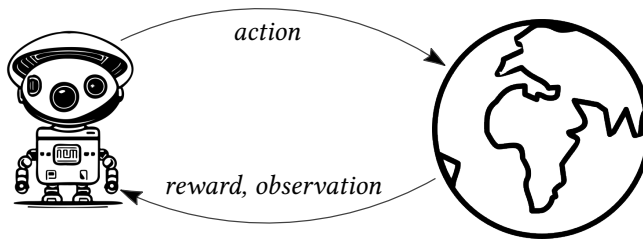


Figure 1.1: The reinforcement learning loop. At each step of the interaction, the agent performs an action, receives a reward, and observes the outcome of the action.

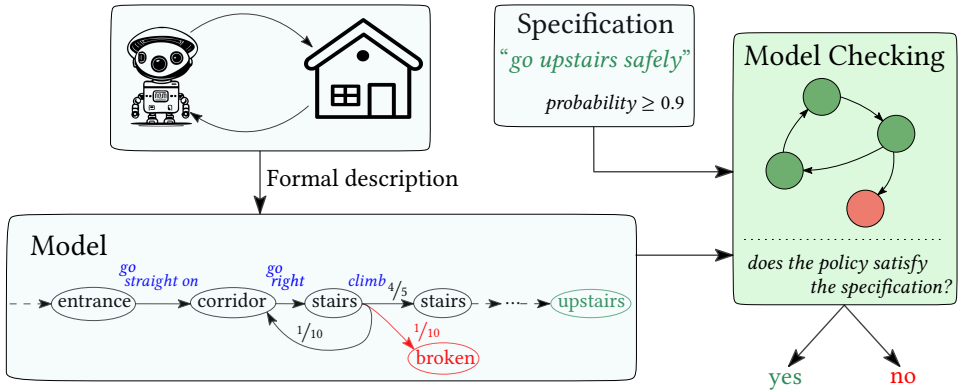


Figure 1.2: Model checking.

learn a representation of the input data, which is crucial to handle high-dimensional spaces.

So, the first question is: **how to ensure that the robot navigates safely and achieves its target objective in complex environments when its control policy has been learned via deep RL?** Taking a step back, it can be useful to consider mathematical techniques that provide guarantees about the correctness of systems.

Formal verification. Now, assume that you have a formal description of the environment and the policy defining the agent’s behavior. Following this description, you may *model* the interaction between the agent and the environment as a “transition system”. You have also formalized the specifications of your agent: “the robot should not collide with obstacles” and “the robot eventually reaches the target location”. Then, in that case, you may apply *model checking* [Baier and Katoen, 2008], which is a formal verification technique that consists as follows. Given a model describing the interaction of the agent with the environment and the properties that the agent should satisfy or not, *model checking* consists in *automatically verifying* if the properties hold in the input model (Figure 1.2).

While model checking is a powerful tool that provides guarantees about the agent’s behavior, it typically relies on an exhaustive exploration of the state space, which makes the technique prone to the *curse of dimensionality*. We discussed it before, your environment is complex, dynamic, uncertain, and high-dimensional. Then, standard model checking tools will usually fail to scale to such environments. Furthermore, the environment is seldom explicitly accessible: you solely designed your agent to *interact*

with the environment, meaning that it is able to execute actions and observe its outcomes, but it does not have access to an explicit description of the environment.

This raises a second question: **how to leverage model checking to verify if the deep RL policy learned in a complex environment satisfies the specifications defined for the agent?**

To answer those questions, I advance the following thesis.

Thesis

By incorporating principles from formal methods, deep reinforcement learning, representation learning, generative modeling, and bisimulation theory, it is possible to *enable model checking policies obtained via deep reinforcement learning*.

Contributions

Markov models, RL, model checking, and bisimulation: they are the four main pillars of this thesis. **Chapter 2** provides a *coherent and comprehensive overview of these concepts*. The contribution lies in their presentation and the way they are unified to further form a consistent framework that allows reasoning about the questions raised above. While RL and model checking have already been discussed, *Markov models* give rise to a mathematical theory that can be used to model the environment, the agent's behavior, and the interaction between the two. Notably, a critical challenge is the choice of *specification formalism* to express the agent's requirements, ensuring compliance with Markov models and RL algorithms. Finally, *bisimulation* is a mathematical concept that allows to relate the behavior of the agent operating within different systems [Larsen and Skou, 1991; Givan et al., 2003]. The concept is central to this thesis, as it allows considering smaller models than the original environment, or abstracted from it, which are amenable for formal verification.

Distillation with guarantees. The core contribution outlined in this manuscript lies midway between RL and formal verification and consists in *the development of the foundations required to enable formal verification of deep RL policies* so that one can leverage the strengths of deep RL to handle complex environments while integrating formal verification to provide guarantees on the agent's behaviors (Figure 1.3). **Chapter 3** covers this proposition, which involves learning an *abstraction* of the environment from the data collected by executing the deep RL policy, resulting in an explicit and tractable model, that we call *latent model*. In parallel, the deep RL policy is *distilled* over the spaces of the latent model, yielding a *latent policy*. *The two are amenable for model checking*. Furthermore, by design, the latent policy can be executed both in the latent model and

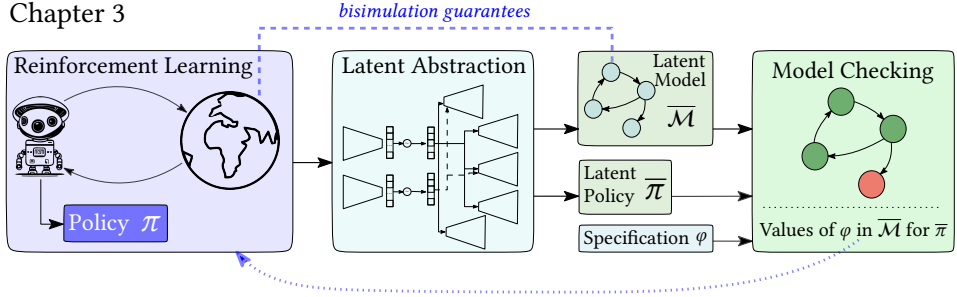


Figure 1.3: Abstraction of the environment into a *tractable* latent model $\bar{\mathcal{M}}$ and distillation of the deep RL policy π over the abstracted, latent spaces, yielding a latent policy $\bar{\pi}$. The latent model $\bar{\mathcal{M}}$ and the distilled policy $\bar{\pi}$ are amenable for model checking. Both are endowed with bisimulation guarantees, which allow to lift the verification results to the original environment. The approach is detailed in **Chapter 3**.

the original environment. The guarantees obtained by verifying the latent model are spit out in the form of *values* that indicate the satisfaction of the input specification in the latent model. The abstraction and distillation processes are further designed to ensure that the latent model obtained is *bisimilar* to the original environment. In our context, bisimulation finally enables lifting the verification results (the *values*) obtained in the latent model to the original environment.

Representation learning and partial observability. Beyond distillation, the bisimulation guarantees obtained in the latent model are also useful for learning representations in the context of deep RL. For instance, you may be interested in learning a representation of the input data that removes the noise and irrelevant information and keeps only a sufficient information for learning the policy. If your robot is equipped with a camera, the raw images may contain irrelevant information for the task at hand, such as the color of the walls or the shadows. Two objects recorded under different angles may look very different in the raw images, while they are actually the same object.

We prove in Chapter 3 that the abstraction procedure groups states from which the agent behaves similarly. Then, the induced representation can be used to facilitate the learning of the policy. In **Chapter 4**, we push this idea further in scenarios where representation learning is crucial: *partially observable environments*. Getting back to your problem of designing a robot, the latter may have noisy sensors that do not provide a full view of the environment. Such sensors might be noisy to the point that the agent is not able to infer directly the true state of the environment from its current observation — such a setting is actually widely encountered in practice.

Chapter 4

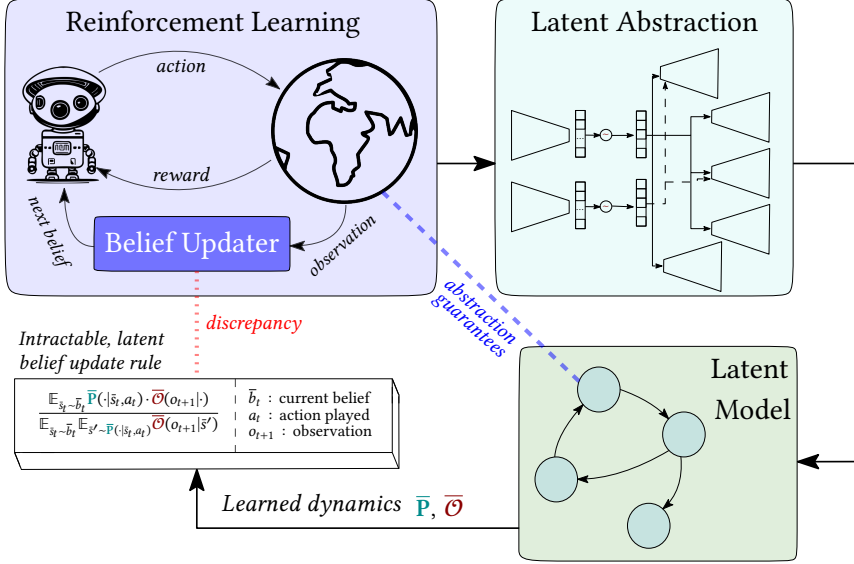


Figure 1.4: Overview of the approach developed in **Chapter 4** for partially observable environments. During the interaction, a latent model is learned with abstraction quality guarantees. While this latent model is now accessible, the belief update rule is still intractable to compute. Then, the dynamics of the latent model are leveraged to learn a “Belief Updater”, which is trained to replicate the intractable belief update rule. The updater allows the agent to maintain a belief over the possible states of the environment at each interaction step. The policy is learned on the induced belief space.

To address this issue, the agent may maintain a *belief* over the possible states of the environment at each interaction step, which is proven to be a sufficient statistic to behave optimally under partial observability [Kaelbling et al., 1998]. However, maintaining this belief requires a model of the environment and is also generally intractable to compute exactly at each step. We build upon the latent model developed in Chapter 3 to provide a framework where learning to maintain a belief is made possible by leveraging the abstraction (Figure 1.4). The result is also guaranteed to be an accurate representation of the original, partially observable, environment. Then, this enables the agent to learn a policy that acts on the belief space learned. We further provide representation guarantees on the belief maintained at each time step.

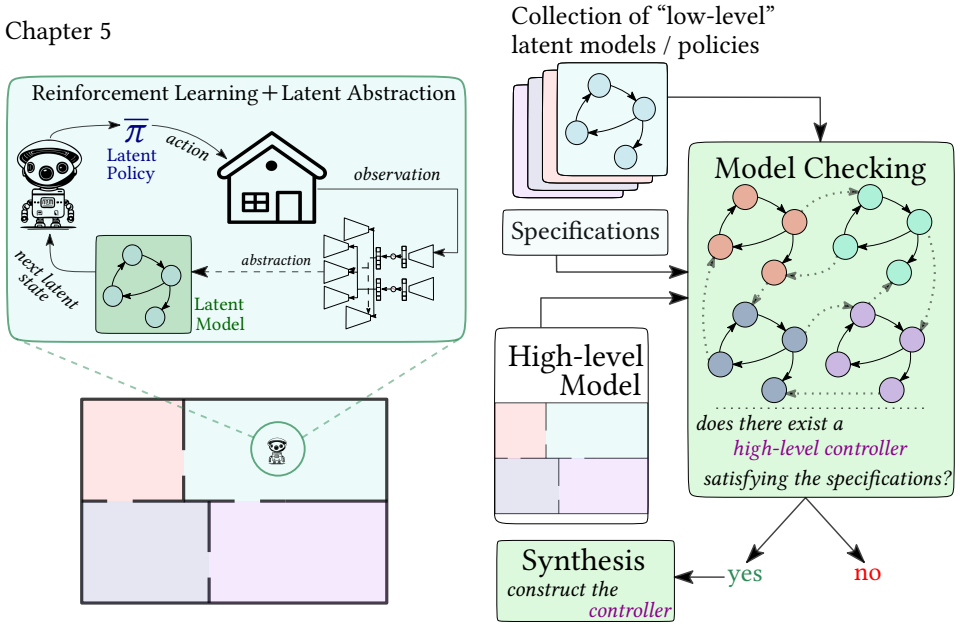


Figure 1.5: Synthesis from deep RL components, presented in **Chapter 5**. The agent learns diverse skills and behaviors across different *rooms*, through verifiable latent models and policies. Then, given (i) a high-level description of the environment, (ii) a collection of low-level models and policies for each room, and (iii) the specifications, the synthesis procedure constructs a *high-level controller* that is guaranteed to satisfy the specification.

Synthesis from deep RL components. Let us get back to the problem of designing a policy for your robot. Since you have now access to a set of tools for providing the agent with formally verified RL policies, you have taught it diverse skills and behaviors that allow it to navigate safely in different rooms and reach some points of interest in the house. However, you may want to go further and design a control policy to achieve more complex tasks, such as grabbing a book from the library, starting from anywhere in the house, and bringing it to the living room. Learning this behavior from scratch is challenging because you need to provide a reward signal that guides the learning process toward achieving this specific task. The most obvious approach would be to provide a single reward only when the book is in the living room. However, this alone is clearly not sufficient to make the robot learn the complex behavior you desire. Remember, you taught the robot to navigate through diverse rooms in the house. Then, a natural approach would be to leverage the skills the robot already knows to achieve the new task.

Chapter 5 shows how the components learned via our latent space models facilitate *synthesis* in very large environments that are typically intractable, even for an RL agent. An overview of the approach is given in Figure 1.5. *Reactive synthesis* refers to techniques constructing a policy in a given model of the environment that satisfies a given objective (typically specified as a logic formula) by design. In fact, one can see synthesis as a step further than model checking, *when no policy is provided*: the model checker then explores the model to decide whether a policy that satisfies the input specification exists in that model. If the answer is “yes”, the synthesis problem requires explicitly constructing such a policy — usually from the results obtained during the model checking procedure. Synthesis thus suffers from the same drawbacks as model checking: it is computationally expensive, struggles to scale complex to environments, and requires an explicit model.

We discussed above the difficulty of modeling the environment explicitly. However, in realistic scenarios, you may be able to provide the agent with some high-level description of the environment. For example, while it may be challenging to detail all the possible interactions the agent may have in your house, you can still provide its plans, the location of the doors, etc. The question is then how to leverage the set of different components (latent models and policies) learned by the agent, the high-level description of the environment, and the specifications to automatically generate a *high-level controller* that is guaranteed to satisfy the input specifications. This further comes with several challenges. Notably, each component learned by the agent may have been learned in different latent spaces, they come with their own set of guarantees, and they may not be directly compatible with the high-level description of the environment. Our approach is designed to address these challenges and provide a solution to this synthesis problem.

Finally, we conclude this thesis in **Chapter 6** by discussing the perspectives and future directions that arise from the contributions presented in this manuscript.

Reading Guide

To aid your navigation through this manuscript, here is a brief reading guide to help you find the sections that are most relevant to you.

Should I really spend some time on the background? How to read it?

This manuscript aims to be accessible to a diverse audience, including readers without prior experience in formal methods or reinforcement learning. A particular effort has been made to provide a comprehensive **background (Chapter 2)**, bridging the gap between the two fields. It not only introduces the necessary notations and knowledge

to understand the contributions but also integrates model checking and reinforcement learning concepts into a coherent framework.

If you are not familiar with the concepts of stochastic model checking and RL, I recommend reading the entire background chapter. Otherwise, you should be familiar with the concept of *Markov decision processes* (MDPs), so be sure to check the notations. Then, you may be tempted to jump directly to Section 2, but I recommend having a look at **Section 1.4** about the **limiting behaviors** that the agent may have within MDPs, especially if you are not a formal methods enthusiast or if you are not familiar with the concepts of *stationary distributions* and *ergodicity*.

Section 2 provides an **introduction to deep reinforcement learning**, covering the main paradigms and algorithms used in practice, the guarantees they provide, and a discussion on the reliability of the learned policies. If you are an RL practitioner, you may be tempted to skip this section, but before, I recommend having a look at the part related to **episodic processes**, in **Section 2.1**. The *episodic setting* is one of the primary (if not the main) settings in which RL is considered, and it comes with a collection of specificities and very useful properties that will be used throughout this manuscript.

Section 3 introduces the main concepts of **stochastic model checking and synthesis**, backed up by the usual linear temporal logic (LTL) formalism. If you are not familiar with the concepts behind model checking, I recommend reading this section entirely. When considering LTL in the context of RL, one faces the fact that some **properties expressible in LTL are not learnable** by RL algorithms in general, which is discussed in **Section 3.4**. Thus, one needs to wisely choose the logic to express the specifications. In this manuscript, we instantiate the *discounted μ -calculus* theory of de Alfaro et al. [2003] in the context of MDPs, giving rise to a *learnable* specification formalism that we call **valued lambda calculus** (VLC). The logic is covered in **Section 3.5**. Discounting in formal verification is not necessarily a common practice (in contrast to RL), and comes with its own set of specificities and challenges, so I recommend reading this section, especially if you never considered discounting in the context of model checking before.

Finally, **Section 4** introduces the concept of **bisimulation** and its derived *distance*, which are central to the contributions of this manuscript. Be sure to check this section as the (same) distance notion has been introduced in two different and complementary ways in the two communities [de Alfaro et al., 2003; Desharnais et al., 2004; Ferns et al., 2005; Chatterjee et al., 2010; Ferns et al., 2011], and we provide a unified view of the two.

I already trained my agent via my favorite RL algorithm and I am mostly interested in formally verifying its behavior.

Then, go to **Chapter 3**. The first part (Section 1) covers the abstraction and representation quality guarantees that can be obtained through a latent space model.

The second and third parts (Sections 2 and 3) cover the *activation* of formal reasoning: we detail how the latent abstraction can be learned through *generative auto-encoders*, and how the deep RL policy can be distilled over the resulting latent model. Once enabled, you may use your favorite model checking tool to verify the distilled policy.

I am interested in training my agent within partially observable environments and I want to have learning guarantees.

Then, go to **Chapter 4**. The framework developed is backed up by the latent space models developed in Chapter 3 and provides a solution to learn a suitable representation of input sequences when the observation is non-Markovian. Compared to state-of-the-art RL algorithms in partially observable environments, we bypass the need for training *recurrent neural networks* to process interaction sequences, which do not provide any learning guarantees and are sometimes intricate to train. Our approach further spits out an abstract model of the environment which enables formal reasoning.

Representation learning is my main concern, I desire to train my agent to learn a representation of the environment that is useful for further learning its policy.

Then, you may want to dive into bisimulation theory. **Chapter 2, Section 4** provides a gentle introduction to the concept. Afterward, you may be interested in the representation guarantees yielded by our latent space models (**Chapter 3, Section 1.2**). Notably, the abstraction procedure described in **Chapter 3, Section 3** is not only useful in the context of distillation, but yields a general latent space learner for RL agents. A deep RL algorithm derived from this idea is further detailed in **Chapter 5, Section 2.2**. Finally, you may want to read **Chapter 4** to see how the latent space model can be used to learn a representation of the input data in partially observable environments, being challenging for standard RL algorithms.

My area of interest is synthesis, but the environment I am considering is too large and intractable for standard verification techniques.

Then, I propose to read **Chapter 1, Section 2** to get familiar with the main concepts of RL and **Chapter 3** with the distillation procedure. This way, you will be able to train deep RL algorithms and then distill the learned policies over a verifiable latent model. This allows checking properties of interest and if the reward signal you provided to the training algorithm is sufficient to learn the desired behavior. More dynamically, you may use the approach developed in Section 3 to combine deep RL and model checking (as presented in **Chapter 5, Section 2.2**) to monitor the satisfaction of the specifications during the learning process.

Finally, if the specification you consider is too intricate or if the environment is too large, even for standard RL techniques, you may want to read **Chapter 5** to see how to

decompose the problem into tractable sub-procedures.

If you lose track of a concept, please don't forget to refer to the index at the end of this manuscript.

Overall, I hope this manuscript provides you with new perspectives on reinforcement learning and formal verification, as well as the potential benefits of combining the two fields. I hope you enjoy reading it.

1 Decision Making under Uncertainty

Decision-making problems involve scenarios where an entity, which we formally denote as the *agent* in this thesis, interacts within an environment intending to fulfill its design objectives. The formalization of this interaction can take various shapes, according to the specific assumptions made about this environment. Questions regarding the nature of this interaction may include whether time proceeds in rounds or elapses continuously, whether the focus lies on worst-case behavioral scenarios for the agent or rather on its expected outcome, and whether the scenario involves a single or multiple agents, engaged in mutual interactions.

Throughout this thesis, we formalize sequential decision-making in uncertain environments as *Markov decision processes* (MDPs). An MDP is a model that embodies both *stochasticity* and *nondeterminism*. Within this framework, the agent engages with a probabilistic environment by realizing (nondeterministic) choices and observing their inherently probabilistic outcomes. Before formally framing the decision-making problem through MDPs, we introduce some concepts from measure theory. This introduction is necessary to get a clear understanding of the underlying concepts of stochasticity, inherent in Markov models.

In this chapter, we may introduce new results via Lemmas and Theorems. To distinguish them from other results, we mark them with the symbol $\star$.

1.1 A Brief Tour through Probability Spaces

In this subsection, we introduce the notions of σ -algebra, probability space, random variable, the expected value of a random variable, and conditional probabilities.

Definition 1: σ -algebra

Let $\mathcal{X}$ be a set, denote by $2^{\mathcal{X}}$ its power set, and let $\Sigma \subseteq 2^{\mathcal{X}}$. Then, Σ is a σ -algebra and the tuple $\langle \mathcal{X}, \Sigma \rangle$ a *measurable space* if and only if Σ satisfies the following properties:

- Σ contains the empty set as well as $\mathcal{X}$: $\emptyset \in \Sigma$ and $\mathcal{X} \in \Sigma$;
- Σ is closed under complementation: if E belongs to Σ , then so does its complement $\mathcal{X} \setminus E$; and
- Σ is closed under *countable unions*: if $E_0, E_1, E_2, \dots$ belong to Σ , then so does $E_0 \cup E_1 \cup E_2 \cup \dots$.

$\mathcal{X}$ is the *sample space*, whose elements are called the *outcomes*, and those of Σ the *events*.

Definition 2: Probability space

Let $\langle \mathcal{X}, \Sigma \rangle$ be a measurable space and write $\mathbb{R}_{\geq 0} = \{x \in \mathbb{R} : x \geq 0\}$ for the set of positive real numbers, then a function $P : \Sigma \rightarrow \mathbb{R}_{\geq 0} \cup \{\infty\}$ is a *probability measure* over $\langle \mathcal{X}, \Sigma \rangle$ if and only if it satisfies the following properties:

- $P(\emptyset) = 0$ and $P(\mathcal{X}) = 1$;
- P maintains countable additivity: for any countable collection $\{E_n\}_{n \in \mathbb{N}}$ of pairwise disjoint events $E_n \in \Sigma$ with $N \subseteq \mathbb{N}$, $P(\bigcup_{n \in \mathbb{N}} E_n) = \sum_{n \in \mathbb{N}} P(E_n)$.

In that case, $\langle \mathcal{X}, \Sigma, P \rangle$ is a *probability space*.

Notations. Given a probability space $\langle \mathcal{X}, \Sigma, P \rangle$, we say that the event $E \subseteq \mathcal{X}$ is *measurable* if E is an element of Σ whose probability is given by $P(E)$. When the measurable space is clear from the context, we write $\Sigma(\mathcal{X})$ for the set of measurable events of $\mathcal{X}$ and $\Delta(\mathcal{X})$ for the set of probability measures over the measurable space $\langle \mathcal{X}, \Sigma \rangle$. A measure P is *deterministic* if it is *Dirac*, i.e., P assigns a value of 1 to any set containing a specific outcome, and 0 to all other sets: there exists an element $x \in \mathcal{X}$ so that $P(E) = 1$ if and only if $x \in E$, for all $E \in \Sigma$. We may use the notation $P = \delta_x$ for any such measure. For the sake of presentation, we may simplify the notation by writing $P(x)$ instead of $P(\{x\})$ when it is clear from the context that $E = \{x\} \subseteq \mathcal{X}$ is a measurable event consisting of a singleton. Finally, we say that an event *almost surely* occurs if it holds with probability one on the probability space, i.e., everywhere with the exception of a set of events with measure zero.

Some examples. Unless otherwise specified, when $\mathcal{X}$ is countable, we assume that $\Sigma = 2^{\mathcal{X}}$. Then, any single outcome is measurable and we use the simplified notation $P(x)$ instead of $P(\{x\})$ for any $P \in \Delta(\mathcal{X})$ and $x \in \mathcal{X}$. We further have $P(E) = \sum_{x \in E} P(x)$ for any $E \in \Sigma$. Moreover, the set $\Delta(\mathcal{X})$ can be seen as the set of *discrete distributions* over $\mathcal{X}$, i.e., $\Delta(\mathcal{X}) = \{P : \mathcal{X} \rightarrow [0, 1] \mid \sum_{x \in \mathcal{X}} P(x) = 1\}$.

Example 1: Dice roll

We consider a simple (fair) dice roll scenario consisting of rolling a pair of dice and observing their resulting uppermost faces. The probability space of a dice roll is $\langle \mathcal{X}, \Sigma, P \rangle$, where:

- the set of outcomes of a dice roll is the finite set $\mathcal{X} = \{\square, \begin{smallmatrix} \square \\ \square \end{smallmatrix}, \begin{smallmatrix} \square \\ \blacksquare \end{smallmatrix}, \begin{smallmatrix} \square \\ \blacksquare \end{smallmatrix}, \begin{smallmatrix} \blacksquare \\ \square \end{smallmatrix}, \begin{smallmatrix} \blacksquare \\ \blacksquare \end{smallmatrix}\}^2$;
- all the possible events are in $2^{\mathcal{X}}$; and
- the probability measure maps every single outcome to $1/36$, i.e., $P(\langle d_1, d_2 \rangle) = 1/36$ for any $\langle d_1, d_2 \rangle \in \mathcal{X}$.

For instance, the probability of obtaining two faces $\square$ is $P(\langle \square, \square \rangle) = 1/36$. The probability of the face of the first dice being $\square$ is in contrast $P(\{\langle \square, d \rangle \mid d \in D\}) = \sum_{d \in D} P(\langle \square, d \rangle) = 6/36 = 1/6$, where $D = \{\square, \begin{smallmatrix} \square \\ \square \end{smallmatrix}, \begin{smallmatrix} \square \\ \blacksquare \end{smallmatrix}, \begin{smallmatrix} \square \\ \blacksquare \end{smallmatrix}, \begin{smallmatrix} \blacksquare \\ \square \end{smallmatrix}, \begin{smallmatrix} \blacksquare \\ \blacksquare \end{smallmatrix}\}$, because the events $\{\langle \square, \square \rangle\}$, $\{\langle \square, \begin{smallmatrix} \square \\ \square \end{smallmatrix} \rangle\}$, $\dots$, $\{\langle \square, \begin{smallmatrix} \blacksquare \\ \blacksquare \end{smallmatrix} \rangle\}$ are all pairwise disjoint.

Other important examples include the *Borel σ -algebras* defined over topological spaces, which are generated by the open sets, i.e., by considering the countable union and relative complement of all intervals in this space. As a concrete example, the Borel σ -algebra on the set of real numbers is the standard general algebra considered in probability theory when dealing with real numbers.

For any discrete or Borel probability space $\langle \mathcal{X}, \Sigma, P \rangle$, we define the *support* of P , written $\text{supp}(P)$, as the smallest closed set of outcomes with non-zero probability: $\text{supp}(P)$ is the smallest closed set $H \subseteq \mathcal{X}$ so that $P(\mathcal{X} \setminus H) = 0$.

The goals of the agent are often formalized as quantities or real-valued signals to optimize. Then, we need to associate each possible outcome with such a quantity. This notion is formally called a *random variable*.

Definition 3: Random variable

Let $\langle \mathcal{X}, \Sigma, P \rangle$ be a probability space and $\mathcal{Y}$ be a set equipped with a σ -algebra, then a *random variable* is a *measurable function* $X : \mathcal{X} \rightarrow \mathcal{Y}$ that maps any possible outcome to a value of $\mathcal{Y}$ so that the pre-image of every measurable subset $I \in \Sigma(\mathcal{Y})$ is measurable as well, i.e., $X^{-1}(I) = \{x \in \mathcal{X} \mid X(x) \in I\} \in \Sigma(\mathcal{X})$.

For $P \in \Delta(\mathcal{X})$, the probability that X takes on a value in the measurable subset $I \in \Sigma(\mathcal{Y})$ is

$$P(X \in I) = P(\{x \in \mathcal{X} \mid X(x) \in I\}).$$

Furthermore, X has for *probability density function* $f_X : \mathcal{Y} \rightarrow \mathbb{R}_{\geq 0} \cup \{\infty\}$ if and only if, given any measurable subset $I \in \Sigma(\mathcal{Y})$, $P(X \in I) = \int_I f_X(y) dy$. We say that P *admits p as density function* (or more simply, P *has density p*) if p is a density for the random variable X consisting in the identity function $X(x) = x$ for all $x \in \mathcal{X}$.

Of particular interest is the average value that X can take according to P .

Definition 4: Expected value

Let X be a real valued random variable over the probability space $\langle \mathcal{X}, \Sigma, P \rangle$, the *expected value of X* is defined as

$$\mathbb{E}_{x \sim P} [X] = \int_{\mathcal{X}} X(x) dP(x).$$

Here, we use the notation $x \sim P$ for *sampling*, or *drawing* an outcome x from the sample space $\mathcal{X}$ according to the probability measure P . Then, $\mathbb{E}_{x \sim P}$ explicitly indicates that the expectation is with respect to the samples x , which are distributed according to P .¹ Notice that the expectation becomes merely a weighted sum in the discrete case: assume that $\mathcal{X}$ is countable, then $\mathbb{E}_{x \sim P} [X] = \sum_{x \in \mathcal{X}} X(x) \cdot P(x)$. If $X : \mathcal{X} \rightarrow \mathcal{Y}$ has for density f_X , we have $\mathbb{E}_{x \sim P} [X] = \int_{\mathcal{Y}} y \cdot f_X(y) dy$. If P admits p as density, then $\mathbb{E}_{x \sim P} [X] = \int_{\mathcal{X}} X(x) \cdot p(x) dx$. We may omit the subscript when it is clear from the context.

Finally, the *variance* $\mathbb{V}_{x \sim P}$ of the random variable X is the squared deviation from the expected value of the random variable:

$$\mathbb{V}_{x \sim P} [X] = \mathbb{E}_{x \sim P} \left[\left(X - \mathbb{E}_{x \sim P} [X] \right)^2 \right] = \mathbb{E}_{x \sim P} [X^2] - \mathbb{E}_{x \sim P} [X]^2.$$

Example 2: The random variable of a dice roll

Consider the probability space $\langle \mathcal{X}, \Sigma, P \rangle$ from Example 1. We define the random variable $X : \mathcal{X} \rightarrow \{2 \leq i \leq 12 : i \in \mathbb{N}\}$ as a mapping from $\mathcal{X}$ to the sum of the numbers depicted on the face of the two dices: $\mathcal{X} = \{ \langle \square, \square \rangle \mapsto 2, \langle \square, \square \rangle \mapsto 3, \dots, \langle \blacksquare, \blacksquare \rangle \mapsto 12 \}$. For instance, the probability of obtaining a roll dice greater than or equal to 11 is $P(X \geq 11) = 1/12$ (via the event $\{ \langle \boxtimes, \blacksquare \rangle, \langle \blacksquare, \boxtimes \rangle, \langle \blacksquare, \blacksquare \rangle \}$).

¹this notation is particularly useful when dealing with complex models with possibly multiple sources of randomness.

Furthermore, the expected value of X is

$$\begin{aligned}
 \mathbb{E}_{\langle d_1, d_2 \rangle \sim P} [X] &= \sum_{d_1, d_2 \in \mathcal{X}} X(d_1, d_2) \cdot P(d_1, d_2) \\
 &= X(\langle \square, \square \rangle) \cdot 1/36 + X(\langle \square, \sqcup \rangle) \cdot 1/36 + \dots + X(\langle \boxplus, \boxplus \rangle) \cdot 1/36 \\
 &= 2 \cdot 1/36 + 3 \cdot 2/36 + 4 \cdot 3/36 + 5 \cdot 4/36 + 6 \cdot 5/36 \\
 &\quad + 7 \cdot 6/36 + 8 \cdot 5/36 + 9 \cdot 4/36 + 10 \cdot 3/36 + 11 \cdot 2/36 + 12 \cdot 1/36 \\
 &= 7.
 \end{aligned}$$

Conditional probabilities. We conclude this tour with the notion of conditioning in the context of probability measures. Let $\langle \mathcal{X}, \Sigma \rangle$ be a measurable space, $\mathcal{Y}$ be another set, and the mapping $P : \mathcal{Y} \rightarrow \Delta(\mathcal{X})$, we use the notation $P(A \mid y)$ as shorthand for $P(y)(A)$ with $A \in \Sigma$, $y \in \mathcal{Y}$. Given any probability measure $P \in \Delta(\mathcal{X})$, the *conditional probability measure* $P : \Sigma \rightarrow \Delta(\mathcal{X})$ is defined through the axiom of probability as

$$P(A \cap B) = P(A \mid B) \cdot P(B),$$

for all $A, B \in \Sigma$ so that $P(B) > 0$. This intuitively means that the joint probability of events A and B is the probability of event B occurring times the one of A occurring provided that B has occurred. This further yields the so-called *Bayes rule*:

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B \mid A) \cdot P(A)}{P(B)}.$$

Similarly, the *conditional expectation* of a random variable X over $\mathcal{X}$ given an event $A \in \Sigma$ is obtained via $\mathbb{E}_{x \sim P} [X \mid A] = \mathbb{E}_{x \sim P} [\mathbb{1}_{\{x \in A\}} \cdot X] / P(A)$, where $\mathbb{1}_{\{x \in A\}}$ is the indicator function for the event A , yielding 1 if $x \in A$ and 0 otherwise.

1.2 Markov Models

As previously mentioned, we consider environments modeled as Markov decision processes, where the agent interacts through nondeterministic choices, and the outcomes of these choices are inherently probabilistic. The stochastic nature of such models is ruled by the so-called *Markov property*. This property dictates that the probability of transitioning from one state of the environment to another is independent of the history of observations and decisions that led to the current state. Instead, it solely depends on the current state and the agent's present choice.

Formally, an MDP is defined as follows.

Definition 5: Markov Decision Process and Markov Chain

A *Markov Decision Process* (MDP) is a tuple $\mathcal{M} = \langle S, \mathcal{A}, \mathbf{P}, \mathcal{R}, \mathbf{AP}, \ell, s_I, \gamma \rangle$ where

- S is a set of *states*;
- $\mathcal{A}$ is a set of *actions*;
- $\mathbf{P} : S \times \mathcal{A} \rightarrow \Delta(S)$ is a *probability transition function* that maps the current state and action to a probability measure over the next states;
- $\mathcal{R} : S \times \mathcal{A} \rightarrow \mathbb{R}$ is a *reward function* that we assume bounded, i.e., there is a bound $R_{\max} \in \mathbb{R}$ so that for all $s \in S$ and $a \in \mathcal{A}$, $|\mathcal{R}(s, a)| \leq R_{\max}$;
- $\ell : S \rightarrow 2^{\mathbf{AP}}$ is a *labeling function* over a set of *atomic propositions* $\mathbf{AP}$;
- $s_I \in S$ is the *initial state*; and
- $\gamma \in (0, 1)$ is a *discount factor* that indicates the agent's preference between short and long-term outcomes.

If $|\mathcal{A}| = 1$, then $\mathcal{M}$ is called a *Markov chain* (MC).

Atomic propositions and labeling functions purely stand for model checking purposes. Intuitively, they are building blocks to formalize properties and specifications that the agent must fulfill. Sometimes, we may omit $\mathbf{AP}$ and ℓ . In that case, we consider the *natural labeling* of $\mathcal{M}$: $\mathbf{AP} = S$ and $\ell(s) = \{s\}$ for each state $s \in S$. When $\mathcal{M}$ is an MC, we may omit the action space in the definition since $|\mathcal{A}| = 1$, $\mathcal{M}$ being in that case a purely stochastic process. We refer to MDPs with uncountable infinite state or action spaces as *continuous MDPs*. In that case, we assume S and $\mathcal{A}$ are complete separable metric spaces equipped with a Borel σ -algebra and we assume that ℓ has an inverse function so that $\ell^{-1}(A) = \{s \in S \mid A \subseteq \ell(s)\}$ is measurable for any $A \subseteq \mathbf{AP}$.

Example 3: Grid world

Consider the grid world environment of Figure 2.1. In this environment, the agent  seeks to reach the target cell  without being eaten by the adversary . The latter is sleeping , but when  enters the cell located at the adversary's left and tries to move upwards  or downwards , the adversary wakes up and tries to eat the agent. In that case, the agent manages to dodge the adversary with probability $9/10$ but randomly escapes upwards or downwards. The adversary then goes back to sleep. Each time the agent stays in the target cell, it receives a reward of **0.1**. If the adversary eats the agent, then the latter is stuck in a sink state where a (negative) reward of **-0.1** is received whatever the move chosen. Everywhere else, the reward obtained upon the execution of an action is null.

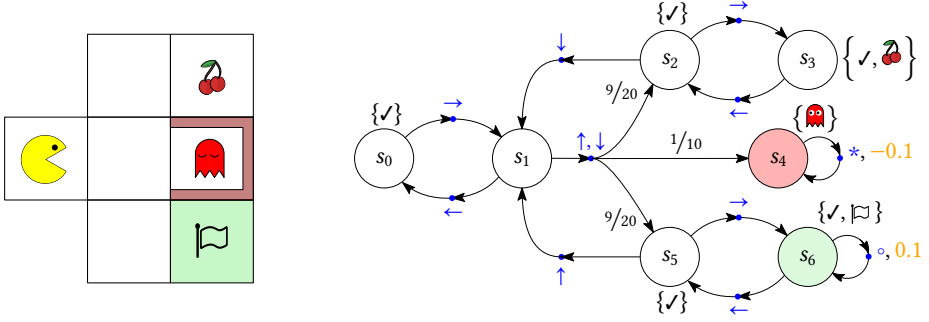


Figure 2.1: Grid world environment (*left*) and its MDP model (*right*). Actions are depicted in blue, labels are drawn next to their states, and non-null rewards are depicted in orange. The symbol $*$ means that the transition holds for all actions. Transitions are labeled by their probability; unlabeled ones are deterministic (probability one). Omitted actions lead to a self-loop (probability one of staying in the current state).

The state space of the related MDP $\mathcal{M}$ is $\mathcal{S} = \{s_0, s_1, \dots, s_6\}$ where the initial state is $s_I = s_0$, the action space is $\mathcal{A} = \{\leftarrow, \rightarrow, \uparrow, \downarrow, \circ\}$, and the atomic propositions are $\text{AP} = \{\checkmark, \boxtimes, \text{ghost}, \text{cherry}\}$. The action $\circ$ represents the *no-op* operation, where the agent does nothing. Every *safe* state is labeled with $\checkmark$.

Intuitively, at each step $t \geq 0$ of the interaction, the agent is in a certain state $s_t \in \mathcal{S}$ of the environment $\mathcal{M}$ and must choose an action $a_t \in \mathcal{A}$. When chosen, the agent executes a_t , then receives a scalar reward $r_t = \mathcal{R}(s_t, a_t)$, and the environment finally evolves to the next state according to the probability distribution defined by the transition function: $s_{t+1} \sim \mathbf{P}(\cdot \mid s_t, a_t)$. The resulting sequence is named a *trajectory* and is defined as follows.

Definition 6: Trajectories and interaction traces

A *trajectory* of $\mathcal{M}$ is a sequence of states and actions $\tau = \langle s_{0:T}, a_{0:T-1} \rangle$ so that $s_0 = s_I$ and $s_{t+1} \sim \mathbf{P}(\cdot \mid s_t, a_t)$ for all $t < T$.^a The sets of *finite* and *infinite* trajectories of $\mathcal{M}$ are respectively denoted by $\text{Traj}_{\mathcal{M}}^{\text{fin}}$ and $\text{Traj}_{\mathcal{M}}$. A *trace* $\hat{\tau} = \langle s_{0:T}, a_{0:T-1}, r_{0:T-1}, l_{0:T} \rangle$ of $\mathcal{M}$ is a finite trajectory where labels and rewards are additionally recorded, i.e., $\langle s_{0:T}, a_{0:T-1} \rangle \in \text{Traj}_{\mathcal{M}}^{\text{fin}}$, $r_t = \mathcal{R}(s_t, a_t)$ for all $t < T$, and $l_t = \ell(s_t)$ for all $t \leq T$. We denote the set of traces of $\mathcal{M}$ of size $T \geq 0$ by $\text{Traces}_{\mathcal{M}}(T)$.

^anotice that a trajectory may consist of a single state by taking $T = 0$.

Example 4: Grid world trajectories

In the grid world of Figure 2.1, $s_0 \xrightarrow{\rightarrow} s_1 \xrightarrow{\downarrow} s_5 \xrightarrow{\rightarrow} s_6$ is a *finite* trajectory, $s_0 \xrightarrow{\rightarrow} s_1 \xrightarrow{\downarrow} s_5 \xrightarrow{\rightarrow} (s_6 \xrightarrow{\rightarrow})^\omega$ is an *infinite* trajectory, and $\langle s_0, \{\checkmark\} \rangle \xrightarrow[0]{\rightarrow} \langle s_1, \emptyset \rangle \xrightarrow[0]{\downarrow} \langle s_4, \{\text{robot}\} \rangle \xrightarrow[-0.1]{\rightarrow} \langle s_4, \{\text{robot}\} \rangle$ is a trace. Here, ω refers to the first ordinal number $\mathbb{N}$ and $(s_6 \xrightarrow{\rightarrow})^\omega$ means that the transition $s_6 \xrightarrow{\rightarrow} s_6$ is repeated infinitely many times.

1.3 Policies

The agent's behaviors in $\mathcal{M}$ are induced by a *policy*, which intuitively plays the role of a template that prescribes to the agent the action to choose at each step of the interaction. Formally, policies are defined as follows.

Definition 7: Policies

A policy $\pi : \text{Traj}_{\mathcal{M}}^{\text{fin}} \rightarrow \Delta(\mathcal{A})$ for $\mathcal{M}$ is a stochastic mapping from sequences of states and actions encountered along the interaction to a probability measure over actions. In particular,

- π is *Markovian* if the decisions solely depend on the system state at the current time step: for all $\tau, \tau' \in \text{Traj}_{\mathcal{M}}^{\text{fin}}$ so that $\tau = \langle s_{0:t}, a_{0:t-1} \rangle$ and $\tau' = \langle s'_{0:t}, a'_{0:t-1} \rangle$ are trajectories of length $t \geq 0$, $s_t = s'_t$ implies $\pi(\cdot | \tau) = \pi(\cdot | \tau')$. Then, π can be defined as the sequence $(\pi_t)_{t \in \mathbb{N}}$ of stochastic mappings $\pi_t : S \rightarrow \Delta(\mathcal{A})$.
- π is *stationary* if the decisions solely depend on the last state of the trajectory: it is a Markovian policy $(\pi_t)_{t \in \mathbb{N}}$ so that $\pi_t(\cdot | s) = \pi_{t+1}(\cdot | s)$ for all $s \in S$ and $t \geq 0$. In that case, π is fully conditioned on states and can be merely defined as the sole stochastic mapping $\pi : S \rightarrow \Delta(\mathcal{A})$.

Property 1: Policies as Mealy machines

Any policy can be encoded via a *stochastic Mealy Machine* $\pi = \langle \mathcal{Q}, \pi_a, \pi_u, q_I \rangle$ where

- $\mathcal{Q}$ is a set of *memory states*;
- $\pi_a : S \times \mathcal{Q} \rightarrow \Delta(\mathcal{A})$ is the *next action function*;
- $\pi_u : S \times \mathcal{Q} \times \mathcal{A} \rightarrow \Delta(\mathcal{Q})$ is the *memory update function*; and
- q_I is the initial memory state.

We say that π has finite memory if $\mathcal{Q}$ is finite and has a memory of size n if $|\mathcal{Q}| = n$.

Again, even when $\mathcal{Q}$ is uncountable, we assume $\mathcal{Q}$ being endowed with a proper σ -algebra. Note that all Markovian policies can be encoded through Mealy machines with countable memory space whose memory update function is solely dependent on the current memory state: $\pi_u(\cdot \mid s, q, a) = \pi_u(\cdot \mid s', q, a)$ for all $s, s' \in \mathcal{S}$, $q \in \mathcal{Q}$, and $a \in \mathcal{A}$. Furthermore, any stationary policy can be encoded through a Mealy machine with $|\mathcal{Q}| = 1$.

Intuitively, the agent acting according to a policy proceeds as follows. At each time step $t \geq 0$, the agent is in state $s_t \in \mathcal{S}$ while its policy π is in state $q_t \in \mathcal{Q}$. The agent executes an action prescribed by π via $a_t \sim \pi_a(\cdot \mid s_t, q_t)$ and the memory of the policy is updated accordingly: $q_{t+1} \sim \pi_u(\cdot \mid s_t, q_t, a_t)$. Then, the agent goes to the next state $s_{t+1} \sim \mathbf{P}(\cdot \mid s_t, a_t)$ and the agent proceeds to the next step. Therefore, policies solve the nondeterminism linked to the choice of actions in $\mathcal{M}$, which leaves us with a purely stochastic process. The aforementioned procedure essentially describes the MC induced by π .

Definition 8: Induced Markov Chain

Let $\pi = \langle \mathcal{Q}, \pi_a, \pi_u, q_I \rangle$ be a policy for $\mathcal{M}$. The *Markov chain induced by π* is

$\mathcal{M}_\pi = \langle S_\pi, \mathbf{P}_\pi, \mathcal{R}_\pi, \mathbf{AP}, \ell_\pi, \langle s_I, q_I \rangle, \gamma \rangle$, where, for all $s \in \mathcal{S}$ and $q \in \mathcal{Q}$, we have

- $S_\pi = \mathcal{S} \times \mathcal{Q}$ is the product of the state space of $\mathcal{M}$ and the memory of π ;
- $\mathbf{P}_\pi(S \times \mathcal{Q} \mid s, q) = \mathbb{E}_{a \sim \pi_a(\cdot \mid s, q)} [\pi_u(Q \mid s, q, a) \cdot \mathbf{P}(S \mid s, a)]$ for $S \in \Sigma(\mathcal{S})$, $Q \in \Sigma(\mathcal{Q})$;
- $\mathcal{R}_\pi(s, q) = \mathbb{E}_{a \sim \pi_a(\cdot \mid s, q)} \mathcal{R}(s, a)$; and
- the labels are the same as in $\mathcal{M}$: $\ell_\pi(s, q) = \ell(s)$.

The transition function embeds the next action and the policy update functions from the policy, and the rewards are averaged over the possible actions produced by the next action function. In the case of stationary policies, we omit the memory dimension since they are memoryless. When considering trajectories in $\mathcal{M}_\pi$, we take the projection of sequence of states, memory, and actions $\langle s_{0:T}, q_{0:T}, a_{0:T-1} \rangle$ of the induced MC to $\mathcal{M}$, i.e., $\langle s_{0:T}, a_{0:T-1} \rangle$.

Example 5: Grid world policies

Consider the MDP from the grid world of Figure 2.1. When the goal of 🟡 is to optimize the accumulated rewards, then the optimal policy is clearly the deterministic stationary policy depicted in Figure 2.2a. Intuitively, this policy brings the agent as fast as possible to the state labeled with 🟡 and prescribes to stay in the latter ad infinitum.

Consider now that the goal of 🟡 is to eat a 🍌 while accumulating rewards. The policy depicted in Figure 2.2b allows to continuously visit both 🍌 and 🟡 until 🟡

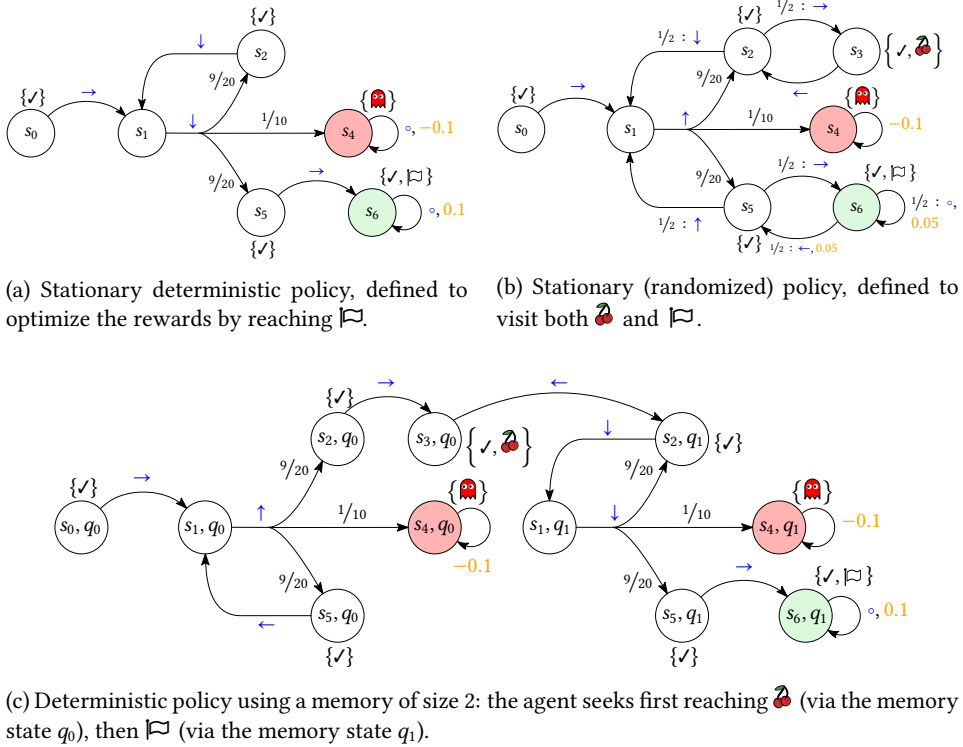


Figure 2.2: Markov chains induced by various policies. Chosen actions are highlighted in blue. Unreachable states are omitted. While the stationary policy highlighted in Figure 2.2b is enough to ensure that 🍌 visits at least once 🍌 , its execution is not safe since states that are not labeled with $\checkmark$ (i.e., s_1 or s_4) are continuously visited (until hitting 🍌). In contrast, the policy highlighted in Figure 2.2c limits the agent visiting s_1 , resulting in safer behavior while allowing collecting more rewards by eventually staying in 🏠 .

eventually catch 🐞. If the agent rather simply cares about visiting 🐞 once, then a safer way to accumulate the rewards is to use a memory. In Figure 2.2c, the policy uses a memory of size 2: the memory is initialized in memory state q_0 where the agent focuses on visiting 🐞 and then, when s_3 is visited, the memory is updated to q_1 where the agent focuses on reaching and staying in s_6 , labeled with 🏠.

Cylinder sets and σ -algebra. Now that we have introduced policies and observed that they induce purely stochastic processes (namely, Markov Chains), it is natural to consider quantifying the agent's performance under a specific policy and measuring its probability of success or failure. To proceed, we require a formal definition of the probability space induced by the policy π in a given MDP $\mathcal{M}$. Precisely, π induces a Markov chain $\mathcal{M}_\pi$ that is equipped with a probability space whose outcomes are trajectories and where *events can be understood as properties*, i.e., conditions that the agent should satisfy or not. The underlying σ -algebra of the induced MC is the one for which the events are generated by the *cylinder sets* spanned by the *prefixes* of trajectories of $\mathcal{M}$. Formally, given a *base* of measurable subsets $B = S_0 \times A_0 \times \dots \times A_{T-1} \times S_T$ so that $A_t \in \Sigma(\mathcal{A})$ and $S_0, S_{t+1} \in \Sigma(\mathcal{S})$ for all $t < T$, the set of trajectory *prefixes* occurring in the scope of the base B is

$$\begin{aligned} \text{Pref}(B) = \Big\{ \langle s_{0:T'}, a_{0:T'-1} \rangle \in \text{Traj}_{\mathcal{M}}^{\text{fin}} \mid \\ T = T', s_0 \in S_0, a_t \in A_t, \\ \text{and } s_{t+1} \in S_{t+1} \text{ for all } t < T \Big\}. \end{aligned}$$

Then, the *cylinder set* spanned by $\text{Pref}(B)$ consists of the set of infinite trajectories that share those prefixes, i.e.,

$$\begin{aligned} \text{Cyl}(B) = \Big\{ \langle s_{0:\infty}, a_{0:\infty} \rangle \in \text{Traj}_{\mathcal{M}} \mid \text{there exists a prefix } \langle s'_{0:T}, a'_{0:T} \rangle \in \text{Pref}(B) \\ \text{with } a_t = a'_t \text{ and } s_{t+1} = s'_{t+1} \text{ for all } t < T \Big\}. \end{aligned}$$

Given a policy π for $\mathcal{M}$, the σ -algebra of $\mathcal{M}_\pi$ is the smallest σ -algebra containing all the cylinder sets $\text{Cyl}(B)$, where the bases B embed the prefixes of $\mathcal{M}$. The outcomes of $\mathcal{M}_\pi$ are $\text{Traj}_{\mathcal{M}}$ while the events $\Sigma(\text{Traj}_{\mathcal{M}})$ are those formed by countable union, intersection, and complementation of cylinder sets. The Carathéodory's extension theorem [Robert B. Ash, 2000] induces a unique probability measure $\mathbb{P}_\pi^{\mathcal{M}}$ over the cylinders sets.

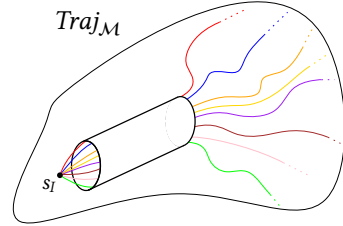


Figure 2.3: Illustration of a cylinder set in the trajectory space. Each colored line corresponds to an infinite trajectory whose prefix is squeezed into a cylinder.

Theorem 1: Probability measure induced by a policy [Revuz, 1984]

Given a policy π for $\mathcal{M}$, $\mathbb{P}_\pi^\mathcal{M}$ is the *unique probability measure of $\mathcal{M}_\pi$* over cylinders spanned by prefixes of measurable bases $B = S_0 \times A_0 \times \dots \times A_{T-1} \times S_T$, with

$$\begin{aligned} \mathbb{P}_\pi^\mathcal{M}(\text{Cyl}(B)) = & \mathbb{1}\{s_I \in S_0\} \int_{A_0} \int_{S_1} \int_Q \pi_a(da_0 \mid s_I, q_I) \cdot \mathbf{P}(ds_1 \mid s_I, a_0) \cdot \pi_u(dq_1 \mid s_I, q_I, a_0) \dots \\ & \int_Q \int_{A_{T-1}} \int_{S_T} \pi_u(dq_{T-1} \mid s_{T-2}, q_{T-2}, a_{T-2}) \cdot \pi_a(da_{T-1} \mid s_{T-1}, q_{T-1}) \cdot \mathbf{P}(ds_T \mid s_{T-1}, a_{T-1}). \end{aligned}$$

We write $\mathbb{E}_\pi^\mathcal{M}$ for the associated expectation operator $\mathbb{E}_{\tau \sim \mathbb{P}_\pi^\mathcal{M}}^\mathcal{M}$, and $\mathbb{P}_\pi^\mathcal{M}(\cdot \mid s_I = s)$ for the measure obtained by replacing s_I by $s \in S$ in $\mathcal{M}$. We omit the superscript when it is clear from the context.

Remark 1: The discrete case

In the discrete context, when the state and action spaces are countable, all states $s \in S$ and actions $a \in \mathcal{A}$ are respectively measurable under $\mathbf{P}$ and π_a , thus it is possible to tighten cylinders to such an extent that the spanned prefix set merely consists of a single finite trajectory [Baier and Katoen, 2008; Baier et al., 2018]. Then, for any $\tau = \langle s_{0:T}, a_{0:T-1} \rangle \in \text{Traj}_\mathcal{M}^{\text{fin}}$,

$$\text{Cyl}(\tau) = \left\{ \langle s'_{0:\infty}, a'_{0:\infty} \rangle \in \text{Traj}_\mathcal{M} \mid a'_t = a_t \text{ and } s'_{t+1} = s_{t+1} \text{ for any } t < T \right\},$$

and

$$\begin{aligned} \mathbb{P}_\pi^\mathcal{M}(\text{Cyl}(\tau)) = & \mathbb{1}\{s_I = s_0\} \mathbb{E}_{q_1 \sim \pi_u(\cdot \mid s_0, q_I, a_0)} \dots \mathbb{E}_{q_{T-1} \sim \pi_u(\cdot \mid s_{T-2}, q_{T-2}, a_{T-2})} \left[\right. \\ & \left. \pi_a(a_0 \mid s_I, q_I) \cdot \mathbf{P}(s_1 \mid s_0, a_0) \dots \pi_a(a_{T-1} \mid s_{T-1}, q_{T-1}) \cdot \mathbf{P}(s_T \mid s_{T-1}, a_{T-1}) \right]. \end{aligned}$$

Consequently, a set of (infinite) trajectories is measurable if and only if it can be derived from a countable union, intersection, or complementation of cylinder sets.

Example 6: Reachability event

In the grid world of Figure 2.1, we define the event of *reaching* $\boxtimes$ as

$$\diamond \boxtimes = \left\{ \langle s_{0:\infty}, a_{0:\infty} \rangle \in \text{Traj}_\mathcal{M} \mid \text{there is a } t \geq 0 \text{ so that } \boxtimes \in \ell(s_t) \right\}.$$

This event is measurable. Indeed, consider the set

$$Pref_{\diamond \sqsupset} = \left\{ \langle s_{0:T}, a_{0:T-1} \rangle \in Traj_{\mathcal{M}}^{fin} \mid \sqsupset \notin \ell(s_t) \text{ for all } t < T \text{ and } \sqsupset \in \ell(s_T) \right\},$$

that embeds every finite trajectory of the form^a

$$s_0 \xrightarrow{\rightarrow} \left(\left(s_1 \xrightarrow{\leftarrow} s_0 \xrightarrow{\rightarrow} \right)^* \left(s_1 \xrightarrow{\downarrow \text{ or } \uparrow} \left(s_2 \xrightarrow{\rightarrow} s_3 \xrightarrow{\leftarrow} \right)^* s_2 \xrightarrow{\downarrow} \right)^* \left(s_1 \xrightarrow{\downarrow \text{ or } \uparrow} s_5 \xrightarrow{\uparrow} \right)^* \right)^* s_1 \xrightarrow{\downarrow} s_5 \xrightarrow{\rightarrow} s_6,$$

where $(\cdot)^*$ means that the content of the parenthesis is repeated finitely many times. Then, it is clear that $\diamond \sqsupset = \bigcup_{\tau \in Pref_{\diamond \sqsupset}} Cyl(\tau)$, which implies its measurability.

Now, let π be the policy depicted in Figure 2.2a. We have by definition of any probability measure that $\mathbb{P}_\pi(\diamond \sqsupset) = \sum_{\tau \in Pref_{\diamond \sqsupset}} \mathbb{P}_\pi(Cyl(\tau))$. Under π , the prefixes that yield a non-zero probability for the event $\diamond \sqsupset$ in $Pref_{\diamond \sqsupset}$ have the form

$$s_0 \xrightarrow{\rightarrow} \left(s_1 \xrightarrow{\downarrow} s_2 \xrightarrow{\downarrow} \right)^* s_1 \xrightarrow{\downarrow} s_5 \xrightarrow{\rightarrow} s_6.$$

Then,

$$\begin{aligned} \mathbb{P}_\pi(\diamond \sqsupset) &= \sum_{i \in \mathbb{N}} \mathbb{P}_\pi \left(Cyl \left(s_0 \xrightarrow{\rightarrow} \left(s_1 \xrightarrow{\downarrow} s_2 \xrightarrow{\downarrow} \right)^i s_1 \xrightarrow{\downarrow} s_5 \xrightarrow{\rightarrow} s_6 \right) \right) \\ &= \sum_{i \in \mathbb{N}} \mathbf{P}_\pi(s_1 \mid s_0) \cdot (\mathbf{P}_\pi(s_2 \mid s_1) \cdot \mathbf{P}_\pi(s_1 \mid s_2))^i \cdot \mathbf{P}_\pi(s_5 \mid s_1) \cdot \mathbf{P}_\pi(s_6 \mid s_5) \\ &= \sum_{i \in \mathbb{N}} \left(\frac{9}{20} \right)^i \cdot \frac{9}{20} = \frac{9/20}{1 - 9/20} = \frac{9}{11}. \end{aligned} \quad (\text{geometric series})$$

^afor the sake of readability, we omitted self-loop transitions occurring with probability one, but they are also part of $Pref_{\diamond \sqsupset}$.

Discussion. Given a policy π for $\mathcal{M}$ and a set of *specifications* formalized as events of $\mathcal{M}_\pi$, *verifying* the behaviors of the agents in $\mathcal{M}$ when it operates under π boils down to checking the probability of those events occurring in $\mathcal{M}_\pi$. The related automated verification procedure is named *model checking* (e.g., Baier and Katoen, 2008; Baier et al., 2018). We may distinguish two kinds of properties: (i) *qualitative* properties, where the events should hold either with probability one or zero and (ii) *quantitative* properties, where the goal is to check whether the events hold with sufficient probability. In this thesis, we rather focus on the latter since they allow for a general analysis of the agent's behaviors. In this context, another typical quantitative analysis is to compute the minimal

or maximal probabilities of achieving the events in $\mathcal{M}$, ranging over all possible policies. This corresponds respectively to the best or worst case analysis of $\mathcal{M}$. Alternatively, we may also start from a specification: in that case, the agent's objective is formalized as optimizing the probability of the given specification. The related automated procedure is named *synthesis* (e.g., Kwiatkowska and Parker, 2013).

As concrete examples, verifying the MDP $\mathcal{M}$ of Example 3 may include the computation of the maximal probability of reaching $\Box$, or the minimal probability of being stuck in states labeled with red robot . In contrast, verifying a policy π for $\mathcal{M}$ amounts checking the probability of events in $\mathcal{M}_\pi$, for instance checking the probability of reaching $\Box$ under the policy of Figure 2.2a, as done in Example 6.

Of course, properties are rarely specified as raw events, i.e., countable union and intersection or complementation of cylinder sets. Instead, they are specified via a formal logic whose semantics allow reasoning about measurable events. We investigate these logics and the algorithms that enable the efficient computation of the induced events in Section 3.

1.4 Limiting Behaviors

We consider now different classes of important properties focused on the *limiting behaviors* exhibited by the agent in $\mathcal{M}$ when it operates under a given policy. Specifically, let π be a stationary² policy for $\mathcal{M}$, those properties are closely related to the following two types of distributions.

- (i) The *transient distribution* $\xi_\pi^t : S \rightarrow \Delta(S)$ maps any state $s \in S$ to the probability of being in a state $s' \in S' \subseteq S$, after *exactly* $t \geq 0$ steps (where S' is a measurable subset of S):

$$\xi_\pi^t(S' | s) = \mathbb{P}_\pi(\{s_{0:\infty}, a_{0:\infty} \mid s_t \in S'\} \mid s_I = s). \quad (2.1)$$

Furthermore, the *occupancy measure* is given by

$$\mu_\pi^t(S' | s) = \frac{1}{t} \sum_{i=1}^t \xi_\pi^i(S' | s) \quad (2.2)$$

and can be interpreted as the fraction of time the agent has spent in the region $S' \in \Sigma(S)$ within t time steps when starting from state $s \in S$.

- (ii) *Stationary distributions* are *invariant distributions* that preserve the behaviors of the agent across transitions and remain unchanged as the time steps increase. Precisely,

²For the sake of presentation, we stick to stationary policies in this subsection. Nevertheless, the results also hold for finite-memory policies by considering the state space of the induced MC $S \times Q$.

for any $S \in \Sigma(S)$, a stationary distribution $\xi_\pi \in \Delta(S)$ satisfies

$$\xi_\pi(S) = \mathbb{E}_{s \sim \xi_\pi} \mathbb{E}_{a \sim \pi(\cdot|s)} \mathbf{P}(S | s, a). \quad (2.3)$$

Then, ξ_π is *invariant* by the transition function $\mathbf{P}$. One can see ξ_π as a distribution over states that are visited when π is executed in a “stable” region of the system: in such a region, the marginal distribution over states visited at any time always corresponds to this stationary distribution. Note that ξ_π need not be unique.

We aim to establish the link between the transient and stationary distributions and to formalize sufficient convergence conditions towards stable regions of the system where a stationary distribution exists. To that aim, we distinctly separate the discrete and the continuous case, where the conditions to reach such stable regions of the system and the properties that hold at the limit of the interaction are different.

The finite case

Assume $\mathcal{M}_\pi$ is finite, i.e., S is finite. In this setting, there exist regions of the system where the agent eventually ends up with probability one and can no longer leave. Those regions are called *bottom strongly connected components* and are defined as follows.

Definition 9: Bottom strongly connected component

Given a stationary policy π for $\mathcal{M}$, a subset $B \subseteq S$ is a *bottom strongly connected component* (BSCC) if and only if the following assertions hold:

- (i) B is *strongly connected*, i.e., there is a finite trajectory linking every pair of states in B : for all states $s, s' \in B$, $\xi_\pi^t(s' | s) > 0$ for some $t \geq 0$;
- (ii) B is a *maximal strongly connected component*, i.e., B cannot be expanded without breaking its strong connectivity; and
- (iii) once B has been reached, B can no longer be left, i.e., $\mathbf{P}_\pi(B | s) = 1$ for all $s \in B$.

Remark 2: BSCCs are linked to policies

It is worth noting that the set of BSCCs of a given MDP $\mathcal{M}$ is linked to the policy executed in $\mathcal{M}$. Concretely, the BSCCs of $\mathcal{M}$ may differ according to the policy considered: any finite MC leads to its own set of BSCCs, and different policies may lead to different MCs, so the set of BSCCs may be different as well.

BSCCs play a crucial role in the analysis of long-term behaviors of agents operating under a given policy. To see why, given an infinite trajectory $\tau = \langle s_0:\infty, a_0:\infty \rangle \in \text{Traj}_{\mathcal{M}_\pi}$,

let us define the set of states visited *infinitely often* as

$$\text{inf}(\tau) = \{s \in S \mid \text{for all } i \geq 0, \text{ there exists } j \geq i \text{ so that } s_j = s\}. \quad (2.4)$$

Note that the event of visiting a set of states $A \subseteq S$ infinitely often is measurable since it can be formalized as countable intersections of unions of cylinders:

$$\{\tau \in \text{Traj}_{\mathcal{M}} \mid \text{inf}(\tau) \cap A \neq \emptyset\} = \bigcap_{i \geq 0} \bigcup_{j \geq i} \bigcup_{\tau \in \text{Pref}_A^j} \text{Cyl}(\tau), \quad (2.5)$$

where Pref_A^j is the set of finite trajectories $\langle s_{0:j}, a_{0:j-1} \rangle \in \text{Traj}_{\mathcal{M}}^{\text{fin}}$ of length j that end up in A with $s_j \in A$. Specifically, the agent is ensured to almost surely end up in a BSCC [Baier and Katoen, 2008], i.e.,

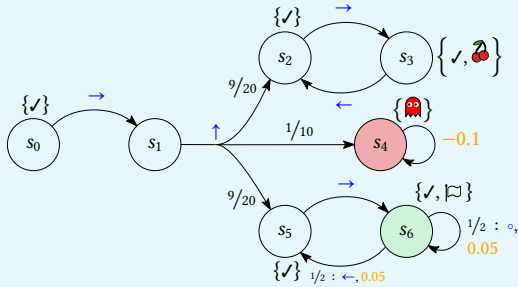
$$\mathbb{P}_{\pi}(\{\tau \in \text{Traj}_{\mathcal{M}_{\pi}} \mid \text{inf}(\tau) \text{ is a BSCC}\}) = 1. \quad (2.6)$$

Stated differently, in every BSCC B induced by the execution of the particular policy π in $\mathcal{M}$, all states of B are almost surely visited infinitely often. Furthermore, each BSCC is equipped with its own unique stationary distribution ξ_{π}^B so that for any pair of states $s, s_{\perp} \in B$, the occupancy measure converges almost surely to ξ_{π}^B :

$$\xi_{\pi}^B(s) = \lim_{t \rightarrow \infty} \mu_{\pi}^t(s \mid s_{\perp}).$$

Example 7: BSCCs in the grid world environment

Let us consider again the grid world environment of Example 3. The MC induced by the deterministic policy of Figure 2.2a has two BSCCs: $\{s_4\}$ and $\{s_6\}$. In contrast, by considering the MC induced by the randomized policy of Figure 2.2b, notice that $\{s_4\}$ is the only BSCC here because s_6 is connected to s_4 : $\xi_{\pi}^t(s_4 \mid s_6) > 0$ when $t \geq 3$. Now, slightly modify the randomized policy of Figure 2.2b so that the agent deterministically goes to s_3 in s_2 and s_6 in s_5 . The resulting policy allows visiting infinitely often $\sqcup$ or $\circ$ since the induced MC has three BSCCs: $B_1 = \{s_2, s_3\}$, $B_2 = \{s_4\}$, and $B_3 = \{s_5, s_6\}$.



Irreducibility and periodicity. If the space of all reachable states of $\mathcal{M}_\pi$ consists of a single BSCC, i.e., if $\{s \in S \mid \text{there exists a } t \in \mathbb{N} \text{ so that } \xi_\pi^t(s \mid s_I) > 0\}$ is a BSCC, then we say that $\mathcal{M}$ is *irreducible* under π . To ensure the convergence to the stationary distribution of the induced MC, we need to investigate the cycles that may occur in the trajectories generated by the agents. In particular, when the agent can return to a given state at specific time points, then the agent never reaches a stable, stationary behavior [Meyn and Tweedie, 1993]. In that case, we say that $\mathcal{M}$ is *periodic* under π . Formally, the *period* of a state $s \in S$ is the greatest common divisor of the set $\{t > 0 \mid \xi_\pi^t(s \mid s) > 0\}$. When the period is one for every state reachable under π , then the induced MC is said *aperiodic*.

Example 8: Periodicity

Take the final policy highlighted in Example 7 with BSCCs B_1, B_2 , and B_3 . Notice that the BSCC B_1 is periodic since the period of both s_2 and s_3 is 2. In contrast, B_2 and B_3 are aperiodic since the period of s_4, s_5 , and s_6 is 1.

Definition 10: Ergodic MDP (finite case)

We say that $\mathcal{M}$ is *ergodic* if and only if $\mathcal{M}_\pi$ is irreducible and aperiodic whatever the stationary policy π for $\mathcal{M}$.

Theorem 2: Convergence to stationarity (finite case)

When $\mathcal{M}_\pi$ is irreducible and aperiodic, then for any $s \in S$,

$$\lim_{t \rightarrow \infty} \xi_\pi^t(s \mid s_I) = \xi_\pi(s), \quad (2.7)$$

where ξ_π is the unique stationary distribution of the unique (reachable) BSCC of $\mathcal{M}_\pi$.

In fact, it is worth noting that Equation 2.7 does not only hold from the initial state s_I , but also from any reachable state under π .

The general case

In the general case, when S is infinite, reasoning about the behaviors of the agent at the limit is considerably more complicated. First, we cannot ensure that the agent almost surely ends up in a region of the environment where states are repeated infinitely often.³

³To see why, simply take a Markov chain where $S = \mathbb{N}$ and $P(i+1 \mid i) = 1$.

Second, there is no notion of *strong* connectivity: when the state space is uncountable, the events that consist in reaching a specific state from any other state are not measurable in general and most of them have zero probability.

For convenience, we focus on a subset of well-behaved Markov chains, namely *Harris chains* (e.g., Athreya and Ney, 1978; Revuz, 1984), which are guaranteed to converge to a stationary distribution at the limit under some conditions. Specifically, we focus on the *recurrent* case, where a measurable region of the system is almost surely visited. This property ensures the existence of a stationary distribution [Durrett, 2010].

Definition 11: Recurrent Harris chain

Let π be a stationary policy for $\mathcal{M}$. Denote the *reachability* event to $H \in \Sigma(S)$ by

$$\diamond H = \{ \langle s_{0:\infty}, a_{0:\infty} \rangle \in \text{Traj}_{\mathcal{M}} \mid \text{there is a } t \in \mathbb{N} \text{ so that } s_t \in H \}.$$

The induced MC $\mathcal{M}_\pi$ is a *recurrent Harris chain* if there exist a measurable set $H \in \Sigma(S)$, an $\epsilon \in (0, 1]$, and a probability measure $\rho \in \Delta(S)$ so that

- (i) H is *recurrent*, i.e., $\mathbb{P}_\pi(\diamond H \mid s_I = s) = 1$ for all $s \in S$; and
- (ii) H is a *small set*, i.e., for any $s \in H$ and $C \in \Sigma(S)$, $\mathbf{P}_\pi(C \mid s) \geq \epsilon \rho(C)$.

Notice that for any $H \in \Sigma(S)$, the event $\diamond H$ is measurable since it can be formalized as a countable union of cylinders:

$$\diamond H = \bigcup_{i \in \mathbb{N}} \text{Cyl}(((S \setminus H) \times \mathcal{A})^{i-1} \times H). \quad (2.8)$$

Then, condition (i) means that the chain almost surely returns to H , from any state $s \in S$. This further implies that the region of the environment defined by H is almost surely visited infinitely often. Condition (ii) means that, when H is reached via *any state* $s \in H$, the next state of $\mathcal{M}_\pi$ can be purely decided according to a biased coin flip with success probability ϵ : if heads, the next state is chosen according to ρ ; if tails (which only occurs when $\epsilon < 1$) the next state is chosen according to the measure defined as $\frac{\mathbf{P}_\pi(C \mid s) - \epsilon \rho(C)}{1 - \epsilon}$, for any $C \in \Sigma(S)$. Importantly, recurrent Harris chains are guaranteed to have a stationary measure ξ_π [Durrett, 2010, Theorem 5.8.9]. In particular,

$$\xi_\pi = \lim_{t \rightarrow \infty} \mu_\pi^t(\cdot \mid s_\perp)$$

almost surely [Asmussen and Glynn, 2011] for any state $s_\perp \in S$.

Example 9: Markov chains with restart

Let $\mathcal{M}$ be any MC with state space S endowed with a Borel σ -algebra. From $\mathcal{M}$, we create another MC $\mathcal{M}'_\varepsilon$ whose state space embeds a special state s_{restart} . A trajectory in $\mathcal{M}'_\varepsilon$ unfolds as follows: at each time step, the process is in a certain state $s \in S$. Then, a Bernoulli-biased coin with success probability $\varepsilon \in (0, 1)$ is tossed. If heads, the Markov chain *restarts*, i.e., it transitions to s_{restart} . Otherwise, if tails, the next state follows the original distribution $s' \sim \mathbf{P}(\cdot \mid s)$. In s_{restart} , the MC deterministically transitions to the initial state s_I of $\mathcal{M}$, and the process starts again. We denote by $\mathbf{P}_\varepsilon$ the resulting transition function.

Processes of this form are referred to as *Markov chains with restart* [Avrachenkov et al., 2013]. These processes find diverse applications in telecommunications and computer science. Notably, protocols like TCP (transmission control protocol) and HTTP (hypertext transfer protocol) exhibit such restarting behavior [Krishnamurthy and Rexford, 2001; Maurer and Huberman, 2001]. Additionally, the GOOGLE PAGERANK algorithm, which simulates the browsing behavior of an Internet user, involves periodic restarts [Brin and Page, 1998].

It is clear that s_{restart} is almost surely visited infinitely often. Therefore, condition (i) from Definition 11 is fulfilled by taking the set $H = \{s_{\text{restart}}\}$. Then, take the measure $\rho \in \Delta(S)$ defined as the Dirac measure $\rho(C) = \mathbb{1}\{s_I \in C\}$, we have that $\mathbf{P}_\varepsilon(C \mid s_{\text{restart}}) = \rho(C)$ for any $C \in \Sigma(S)$, which means that H is a small set and fulfills condition (ii). Therefore, $\mathcal{M}'_\varepsilon$ is a recurrent Harris chain.

Periodicity. Similarly to the discrete case, we say that a Harris chain is *aperiodic* if and only if there is a moment from which the agent can return to the region H *at any time*: for any $s \in H$, there exists a moment $i \in \mathbb{N}$ so that for any $j \geq i$, $\xi_\pi^j(H \mid s) > 0$.

Definition 12: Ergodic MDP (general case)

An MDP that induces an aperiodic recurrent Harris chain under any stationary policy is *ergodic*.

Theorem 3: Convergence to stationarity (general case)

Assume $\mathcal{M}_\pi$ is an aperiodic recurrent Harris chain with stationary distribution ξ_π , then ξ_π^t converges to ξ_π in total variation as t goes to ∞ , i.e.,

$$\lim_{t \rightarrow \infty} d_{TV}(\xi_\pi^t(\cdot \mid s), \xi_\pi) = 0,$$

for any $s \in S$ and where d_{TV} is the *total variation distance*, defined as $d_{TV}(P, Q) = \sup_{A \in \Sigma} |P(A) - Q(A)|$ for any $P, Q \in \Delta(\mathcal{X})$ defined on the σ -algebra $\langle \mathcal{X}, \Sigma \rangle$.

Additional notations. In the sequel, we may write $s \sim \xi_\pi^t$ for $s \sim \xi_\pi^t(\cdot \mid s_I)$ given $t \geq 0$. We also write μ_π^t for $\mu_\pi^t(\cdot \mid s_I)$ and $s, a \sim d_\pi$ for first sampling s from d_π , then a from $\pi(\cdot \mid s)$ with $d \in \{\xi_\pi, \xi_\pi^t, \mu_\pi^t\}$. Similarly, we write $s, a, r \sim d_\pi$ and $s, a, r, s' \sim d_\pi$ where we draw $r = \mathcal{R}(s, a)$ and s' from $\mathbf{P}(\cdot \mid s, a)$, after that s, a have been sampled from d_π .

1.5 Return and Value Functions

Until now, we drafted two kinds of agent objectives. Namely, either the policy is designed to maximize or minimize the probability of satisfying some properties in $\mathcal{M}$. This boils down to finding a policy that induces the probability measure with the highest or lowest probability of satisfying the related events. Alternatively (or in a complementary way), the agent's behavior can be purely reward-driven. In that case, the agent's goal is to maximize the *expected return*, where the *return* is a random variable being defined as the discounted cumulative rewards accumulated along the trajectories.

Definition 13: Expected return

Given a policy π for $\mathcal{M}$ with discount factor $\gamma \in (0, 1)$, the *expected return* of $\mathcal{M}$ is

$$\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \cdot \mathcal{R}(s_t, a_t) \right]. \quad (2.9)$$

Definition 14: Value function

The *value function* is a mapping $V^\pi : \mathcal{S} \rightarrow \mathbb{R}$ that yields the expected return obtained from any state $s \in \mathcal{S}$ by running π in $\mathcal{M}$, i.e.,

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

The typical goal of an agent is to find a policy that optimizes the expected return in $\mathcal{M}$, i.e., $\arg \sup_\pi V^\pi(s_I)$. We denote the *optimal* values of $\mathcal{M}$ by V^* , thus defined as $V^*(s) = \sup_\pi V^\pi(s)$. We further write $Q^*(s, a) = \mathcal{R}(s, a) + \mathbb{E}_{s' \sim \mathbf{P}(\cdot \mid s, a)} [\gamma \cdot V^*(s')]$ for the *Q-value* of the state-action pair $\langle s, a \rangle \in \mathcal{S} \times \mathcal{A}$. In fact, an optimal policy can be obtained by acting purely greedily w.r.t. the value of every state, as described in the following Theorem.

Theorem 4: Bellman's equation [Bellman, 1957]

The optimal value function V^* is the *unique solution* of the Bellman's equation

$$V^*(s) = \sup_{a \in \mathcal{A}} \left[\mathcal{R}(s, a) + \mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, a)} [\gamma \cdot V^*(s')] \right] \quad \text{for all } s \in \mathcal{S}, \quad (2.10)$$

or more compactly $V^*(s) = \sup_{a \in \mathcal{A}} Q^*(s, a)$ for all $s \in \mathcal{S}$.

Notice that in the particular case of MCs, the sup operator of Equation 2.10 vanishes since the action space merely consists of a singleton. Precisely, given any policy π for $\mathcal{M}$, the optimal value function of $\mathcal{M}_\pi$ is the value function V^π where

$$V^\pi(s) = \mathcal{R}_\pi(s) + \mathbb{E}_{s' \sim \mathbf{P}_\pi(\cdot | s)} [\gamma \cdot V^\pi(s')], \quad (2.11)$$

for any state $s \in \mathcal{S}_\pi$. Similarly, we write $Q^\pi(s, a) = \mathcal{R}(s, a) + \mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, a)} [\gamma \cdot V^\pi(s')]$ for the Q -value function of the state-action pair $\langle s, a \rangle \in \mathcal{S}_\pi \times \mathcal{A}$, when the agent operates in $\mathcal{M}$ by following the actions prescribed by π .

Theorem 5: Stationary policies are sufficient [Puterman, 1994]

For all $\epsilon > 0$, we say that a policy π is ϵ -optimal for the expected return if and only if $V^\pi(s) > V^*(s) - \epsilon$ for all $s \in \mathcal{S}$. Then,

- (a) if $\mathcal{A}$ is uncountable and endowed with a Borel σ -algebra, then an ϵ -optimal stationary policy almost surely exists;
- (b) if $\mathcal{A}$ is countable, then there exists an ϵ -optimal stationary policy; and
- (c) if $\mathcal{A}$ is finite, then there exists a *deterministic* optimal stationary policy.

In particular, when $\mathcal{A}$ is finite, let $\pi^*(a | s) = \mathbb{1} \{a = \arg \max_{a' \in \mathcal{A}} Q^*(s, a')\}$, then

$$V_{\pi^*}(s_I) = V^*(s_I) = \sup_{\pi} \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t \cdot \mathcal{R}(s, a) \right].$$

Example 10: Values of the optimal policy in the grid world environment

Let $\mathcal{M}$ be the MDP modeling the grid world environment of our running Example 3. It is easy to notice that the deterministic stationary policy π of Figure 2.2a is an

optimal policy: intuitively, it allows reaching as quickly and safely as possible the sole BSCC that yields positive rewards in $\mathcal{M}$.

The Bellman's equation for the value function V^π (Equation 2.11) leads to the following system of linear equation:

$$\begin{aligned} V^\pi(s_0) &= \gamma \cdot V^\pi(s_1) & V^\pi(s_1) &= \gamma \cdot \left(\frac{9}{20} V^\pi(s_2) + \frac{1}{10} V^\pi(s_4) + \frac{9}{20} V^\pi(s_5) \right) \\ V^\pi(s_2) &= \gamma \cdot V^\pi(s_1) & V^\pi(s_4) &= -\frac{1}{10} + \gamma \cdot V^\pi(s_4) \\ V^\pi(s_5) &= \gamma \cdot V^\pi(s_6) & V^\pi(s_6) &= \frac{1}{10} + \gamma \cdot V^\pi(s_6) \end{aligned}$$

Let us solve the system for $\gamma = 0.9$. We start with the two BSCCs of $\mathcal{M}_\pi$: $V^\pi(s_6) \cdot (1 - \gamma) = \frac{1}{10}$, which means that $V^\pi(s_6) = 1$, and $V^\pi(s_4) = -1$ by the same reasoning. Thus, $V^\pi(s_5) = \gamma = 0.9$. Then, we have:

$$\begin{aligned} V^\pi(s_1) &= \gamma \cdot \left(\frac{9}{20} V^\pi(s_2) + \frac{1}{10} V^\pi(s_4) + \frac{9}{20} V^\pi(s_5) \right) \\ &= \gamma \cdot \left(\frac{9}{20} V^\pi(s_1) - \frac{1}{10} + \frac{81}{200} \right) \\ &= \frac{9\gamma^2}{20} V^\pi(s_1) - \frac{\gamma}{10} + \frac{81\gamma}{200} \\ \iff V^\pi(s_1) \cdot (1 - \frac{9\gamma^2}{20}) &= \frac{81\gamma}{200} - \frac{\gamma}{10} \\ \iff V^\pi(s_1) &= \frac{\frac{81\gamma}{200} - \frac{\gamma}{10}}{1 - \frac{9\gamma^2}{20}} \\ &= \frac{549}{1271}. \end{aligned}$$

Then, $V^\pi(s_0) = V^\pi(s_2) = \frac{549\gamma}{1271} = \frac{4941}{12710} \approx 0.39$.

1.6 Solving MDPs for the Expected Return

Linear Programming

Interestingly, when the state-action space of $\mathcal{M}$ is finite, the problem of computing the optimal value function V^* can be encoded through the following *linear program* (LP) of $|S|$ variables and $|S \times \mathcal{A}|$ constraints, derived from the Bellman's equation [d'Epenoux, 1963]:

$$\begin{aligned} &\textbf{Minimize} && \sum_{s \in S} v_s \\ &\textbf{Subject to} && v_s \geq \mathcal{R}(s, a) + \sum_{s' \in S} \gamma \cdot \mathbf{P}(s' \mid s, a) \cdot v_{s'} \quad \text{for all } s \in S, a \in \mathcal{A}. \end{aligned} \quad (2.12)$$

The unique solution of this LP coincides with the unique solution of the Bellman equation, i.e., $V^*(s) = v_s$ for all $s \in S$. LPs can be solved in polynomial time in the

number of variables, constraints, as well as the maximal number of bits used to represent the coefficients of v , which naturally leads to the following theorem.

Theorem 6: Complexity of solving MDPs

An optimal deterministic stationary policy defined for any finite MDP $\mathcal{M}$ can be constructed in polynomial time in the size of $\mathcal{M}$.

Intuitively, once the solution of the LP is found, optimal policies can be constructed by applying Theorem 5(c). This boils down to finding the actions that tighten the constraints of Equation 2.12.

Discussion. While Theorem 6 is highly significant in theory, the degree of polynomial of the resulting LP is nontrivial, and LP solvers that ensure polynomial-time efficiency run slowly in practice. Alternatively, one can use simplex-based methods [Dantzig, 1991] to solve the LP, which are known to perform well in practice, but their running time is exponential. Nevertheless, it is worth mentioning that when the LP of Equation 2.12 is used to encode the values of a Markov chain $\mathcal{M}_\pi$ to *evaluate* the values V^π of a given policy π for $\mathcal{M}$, all the constraints of Equation 2.12 are tight (there is only one constraint left per state). This means that we are left with the simple system of $|S_\pi|$ linear equations of Equation 2.11, whose degree of polynomial is affordable⁴ (e.g., as in Example 10).

But can we do better? Is it possible to devise an algorithm that runs fast in practice for general MDPs? Importantly, LP solvers do not leverage the MDP structure significantly, in contrast to *iterative methods*, which can be derived from dynamic programming.

Value Iteration

Let $\mathcal{V}$ be the space of bounded⁵ real-valued function $V : S \rightarrow \{x \in \mathbb{R} : |x| \leq R_{\max}/1-\gamma\}$. We define the *Bellman operator* $\mathcal{B} : \mathcal{V} \rightarrow \mathcal{V}$ as

$$(\mathcal{B}V)(s) = \sup_{a \in \mathcal{A}} \left[\mathcal{R}(s, a) + \mathbb{E}_{s' \sim \mathbf{P}(\cdot|s, a)} [\gamma \cdot V(s')] \right]. \quad (2.13)$$

Similarly, let $\mathcal{V}_{\mathcal{A}}$ be space of real-valued functions $Q : S \times \mathcal{A} \rightarrow \{x \in \mathbb{R} : |x| \leq R_{\max}/1-\gamma\}$, by slight abuse of notation, the Bellman operator can also be applied on the $\mathcal{V}_{\mathcal{A}}$ space, where

$$(\mathcal{B}Q)(s, a) = \mathcal{R}(s, a) + \mathbb{E}_{s' \sim \mathbf{P}(\cdot|s, a)} \left[\gamma \cdot \sup_{a' \in \mathcal{A}} Q(s', a') \right]. \quad (2.14)$$

⁴solving a system of n linear equations has a complexity of at most $O(n^3)$, the best-known algorithm running in $O(n^{2.376})$ [Coppersmith and Winograd, 1987].

⁵since $\sup_{s \in S, a \in \mathcal{A}} |\mathcal{R}(s, a)| \leq R_{\max}$, we have $\sup_{s \in S} |V(s)| \leq \sum_{t=0}^{\infty} \gamma^t \cdot R_{\max} = R_{\max}/1-\gamma$.

Then, the Bellman's Equation 2.10 can be rewritten as $V^* = \mathcal{B}V^*$ (resp. $Q^* = \mathcal{B}Q^*$). In particular, for any policy π for $\mathcal{M}$, applying $\mathcal{B}$ in the MC $\mathcal{M}_\pi$ also yields $V^\pi = \mathcal{B}V^\pi$ (resp. $Q^\pi = \mathcal{B}Q^\pi$) since the supremum vanishes in any MC (cf. Equation 2.11). This formulation allows deriving interesting properties from the Bellman operator, which are useful to practically compute the expected return.

Lemma 1: Monotone contraction mapping [Puterman, 1994]

Define the relation $\leq \in \mathcal{V} \times \mathcal{V}$ as $\{\langle V, U \rangle \mid V(s) \leq U(s) \text{ for all } s \in \mathcal{S}\}$ and $\|\cdot\|_\infty$ as the *supremum norm*, i.e., $\|V\|_\infty = \sup_{s \in \mathcal{S}} |V(s)|$ so that $\|V - U\|_\infty = \sup_{s \in \mathcal{S}} |V(s) - U(s)|$ for any $V, U \in \mathcal{V}$. The Bellman operator $\mathcal{B}$ fulfills the following properties:

- (i) $\mathcal{B}$ is *monotone*, i.e., for any $V, U \in \mathcal{V}$, $V \leq U$ implies that $\mathcal{B}V \leq \mathcal{B}U$;
- (ii) $\mathcal{B}$ is a *contraction* under the supremum norm, i.e., $\|\mathcal{B}V - \mathcal{B}U\|_\infty \leq \alpha \|V - U\|_\infty$ for some $\alpha \in [0, 1)$. In particular, $\mathcal{B}$ is a contraction for $\alpha = \gamma$.

Lemma 1 suggests that Equation 2.10 can be approximated via dynamic programming [Bellman, 1957]. In particular, by the Banach's fixed point Theorem (e.g., Puterman, 1994), the sequence $(V_i)_{i \in \mathbb{N}}$ defined as $V_{i+1} = \mathcal{B}V_i$ converges to V^* with linear convergence rate γ . The related (generic) algorithm is called *value iteration* (VI) and is described in Algorithm 1. In its simplest version, the algorithm stops when the largest value difference obtained between two consecutive steps falls below an error term ϵ . Note that $(V_i)_{i \in \mathbb{N}}$ converges whatever the initial values $V_0 \in \mathcal{V}$. In the context of optimizing the expected return, ϵ is often called the *Bellman error*.

Algorithm 1: Value Iteration

Input:

- update function $\mathcal{U} : \mathcal{V} \rightarrow \mathcal{V}$;
- initial values $V_0 \in \mathcal{V}$; and
- error term $\epsilon > 0$.

▷ Initialize the value function

$V \leftarrow V_0$

repeat

$V' \leftarrow V$

 ▷ Update the values

$V \leftarrow \mathcal{U}V'$

until $\|V - V'\|_\infty \leq \epsilon$

return V

Lemma 2: Bellman error [Puterman, 1994]

Let $\epsilon = \|V_{i+1} - V_i\|_\infty$ be the *Bellman error*, then $\|V_{i+1} - V^*\|_\infty \leq \frac{\gamma\epsilon}{1-\gamma}$. Furthermore, when $\mathcal{A}$ is finite with $|\mathcal{A}| > 1$, let π be a deterministic stationary policy so that

$$\pi(a \mid s) = \mathbb{1} \left\{ a = \arg \max_{a' \in \mathcal{A}} \mathcal{R}(s, a') + \mathbb{E}_{s' \sim P(\cdot \mid s, a')} [\gamma \cdot V_{i+1}(s')] \right\},$$

then π is a $(2\gamma\epsilon/(1-\gamma))$ -optimal *greedy* policy.

In consequence, given an error term $\delta > 0$, Lemma 2 suggests that obtaining an approximation with error δ to V^* or V^π can be achieved by setting the Bellman error ϵ to $\delta(1-\gamma)/\gamma$, while further ensuring the 2δ -optimality of the resulting greedy policy. Other types of stopping criteria (e.g., based on relative instead of absolute error) can also be adopted, depending on the initial values, the update function, and the guarantees sought when the execution of the VI algorithm stops [Hartmanns and Kaminski, 2020; Granzotto et al., 2021; Kretínský et al., 2023].

Interestingly, VI with update function $\mathcal{B}$ is guaranteed to converge in norm, even if $\mathcal{S}$ and $\mathcal{A}$ are not finite [Puterman, 1994], essentially because $\mathcal{B}$ is a contraction mapping on $\mathcal{V}$. However, in practice, VI is only generally applicable when $\mathcal{S}$ and $\mathcal{A}$ are finite.

Time complexity. Assuming $\mathcal{M}$ is finite, one can easily notice that each value update can be computed in polynomial time in the size of $\mathcal{M}$, but the number of iterations required to reach the stopping criterion is polynomial in $1/(1-\gamma)$ [Littman et al., 1995]. While VI is still guaranteed to operate in polynomial time assuming γ is *fixed* (i.e., not part of the input), γ is generally assumed to be a rational number given in binary that is part of the instance of $\mathcal{M}$, so the number of value updates required is generally exponential⁶ (see Balaji et al., 2019 for a further discussion).

⁶precisely, it is *pseudo-polynomial*: it would have been polynomial if the encoding of the input was given in unary, but it is exponential since the input is binary encoded. Here, VI is generally exponential in the number of bits required to encode the denominator of γ (assuming γ is a rational number).

Policy Iteration

Policy iteration (PI, Algorithm 2) is another iterative method that rather consists in strictly alternating between *policy evaluation* and *policy improvement* steps. Specifically, we start with a policy π_0 that is initialized by greedily selecting arbitrary actions. At each iteration $i \geq 0$, V^{π_i} is evaluated via the function $\mathbf{eval}(\pi)$, which can be implemented through VI or a linear equation solver. Then the policy is improved by selecting the action that maximizes the current Q -values for the policy π_i , i.e., $\pi_{i+1}(a | s) = \mathbb{1} \{a = \arg \max_{a'} Q^{\pi_i}(s, a')\}$ for every state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$.

Algorithm 2: Policy Iteration

Input:

- evaluation function $\mathbf{eval}(\pi)$;
- initial policy π_0 for $\mathcal{M}$.

▷ Initialize the policy

$\pi \leftarrow \pi_0$

repeat

▷ Policy evaluation

$V \leftarrow \mathbf{eval}(\pi)$

▷ Policy improvement

$\pi' \leftarrow \pi$

foreach s **in** $\mathcal{S}$ **do**

$a^* \leftarrow \arg \max_{a \in \mathcal{A}} \mathcal{R}(s, a) +$

$\mathbb{E}_{s' \sim \mathbf{P}(s, a)} [\gamma \cdot V(s')]$

$\pi(a^* | s) \leftarrow 1$

$\pi(a | s) \leftarrow 0$ for all $a \in \mathcal{A} \setminus \{a^*\}$

until $\pi = \pi'$

return π

Lemma 3: Convergence of policy iteration [Puterman, 1994]

Let π_i be the policy resulting from the i^{th} iteration of PI where $\mathbf{eval}(\pi_i) = V^{\pi_i}$ is implemented as the resolution of the system of linear equations yielded by Equation 2.11 for the MC $\mathcal{M}_{\pi_i}$. Additionally, let $(V_i)_{i \in \mathbb{N}}$ be the sequence resulting from VI when (i) the Bellman operator $\mathcal{B}$ is used as the update function, and (ii) V^{π_0} as the initial values V_0 . Then, for all $i \geq 0$ and $s \in \mathcal{S}$,

$$V_i(s) \leq V^{\pi_i}(s) \leq V^*(s).$$

Since VI converges to V^* , then so does PI.

While Lemma 3 suggests a quicker convergence of PI to the optimal policy in terms of iterations compared to VI, PI is often particularly less efficient. This is because the policy evaluation, performed at each update step, crucially relies on solving the system of linear equations of the MC induced by the current policy, making it drastically more computationally intensive than a straightforward VI update step.

Time complexity. One can always derive an error term $\delta > 0$ for VI that implies the optimality of the resulting greedy policy [Littman et al., 1995; Balaji et al., 2019], and again, the resulting number of iterations $i \in \mathbb{N}$ required is exponential in the size of $\mathcal{M}$ (polynomial in $1/(1-\gamma)$, but γ is a binary encoded rational number). Consequently, by Lemma 3, the running time of PI is still exponential.

Discussion

So far, we presented an LP encoding yielding polynomial solutions, but whose solvers run slowly. At the price of switching to exponential-time solvers (simplex-based methods), faster solutions can be derived, but the resulting algorithms do not leverage the MDP structure. Alternatively, one may craft another exponential time algorithm that iteratively computes the value function through a contraction mapping and returns approximate solutions, while running fast in practice. Finally, we presented another iterative exponential algorithm that delivers exact solutions.

So which algorithm runs faster? It is folklore that VI tends to yield solutions faster, even in the undiscounted case ($\gamma = 1$), where the Bellman operator $\mathcal{B}$ is not a contraction mapping. This is the reason why VI is the default MDP solver in the majority of model checkers [Budde et al., 2020; Hartmanns et al., 2023]. Using VI, all operations can further be implemented through multidimensional (possibly sparse) arrays, enabling the use of efficient numerical solvers for linear algebra (e.g., Bargiacchi et al., 2020; Hensel et al., 2022) which can be parallelized on GPU, yielding solutions even faster (e.g., Delgrange et al., 2023a; Farrington et al., 2023).

While VI and PI notably depend on the exhaustive knowledge of the model to iteratively compute the full value vector, challenges arise when dealing with large state spaces or *sparse rewards* — zero reward for the majority of transitions. In such cases, certain, potentially substantial, parts of the state space contribute minimally to the computation of the expected return from the initial state and exhaustive updates (over the whole state-action space) scale poorly. Additionally, in practice, a model of the environment is seldom explicitly available (i.e., in explicit MDP formalism), and the environment is often very large (or even infinite), which hinders the applicability of dynamic programming. An alternative is to *learn* the optimal value function or policy instead of computing them explicitly, solely based on the interactions of the agent with the environment. This method is named *reinforcement learning* and is detailed in the following section. Even though VI and PI are not applicable when the MDP is not explicitly available, those two approaches will serve as general templates for the algorithms presented in the sequel.

2 Reinforcement Learning

Reinforcement Learning (RL) comprises algorithms that empower an autonomous agent to *learn* a control policy through interaction with an *unknown* environment by gathering experiences in the form of execution traces. In the context of this thesis, we assume that the environment is modeled as an MDP $\mathcal{M}$ (e.g., Szepesvári, 2010). The term *unknown* is pivotal here, indicating that the agent lacks explicit access to $\mathcal{M}$, but can instead interact with it dynamically. Typically, the goal of an RL agent is to maximize its expected return.

The interaction follows the usual RL loop described in Figure 2.4. At each time step $t \geq 0$, the agent executes an action a_t in the environment $\mathcal{M}$, then it receives a scalar reward r_t and observes the next state of the environment s_{t+1} . When receiving this feedback from $\mathcal{M}$, the agent **updates** its current policy π according to the transition $\langle s_t, a_t, r_t, s_{t+1} \rangle$ observed. Then, the next action is drawn from some *behavioral policy* π^b embedding in its memory (some information about) the current policy. Note that the agent *does not know* and *does not get access to* $\mathcal{R}$ nor $\mathbf{P}$: it can only interact with $\mathcal{M}$ to reinforce its policy π from the traces drawn in $\mathcal{M}$ through π^b .

Exploration vs. exploitation. To balance between *exploration* — trying actions to reach unvisited regions of the state space and discover further reward strategies — and *exploitation* — executing the best actions known so far by the agent — the behavioral policy π^b should be wisely chosen. Too much exploration slows down the learning process, while too much exploitation may cause the agent to be stuck in local optima (see, e.g., Amin et al., 2021; Ladosz et al., 2022 for a discussion about the exploration vs. exploitation duality). When π^b coincides with π , we say that the RL algorithm is *on-policy*, since it explores the environment with the actual policy learned. Otherwise, the algorithm is said *off-policy*.

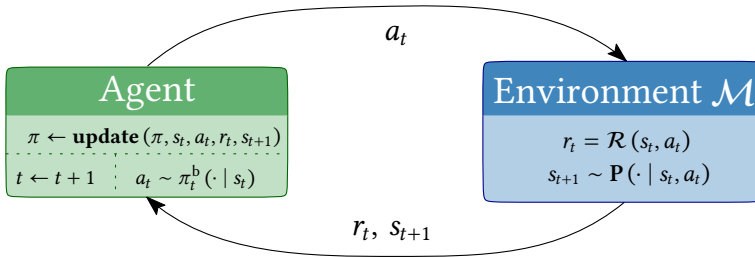


Figure 2.4: Classic reinforcement learning loop.

Optimization of parameterized functions. Henceforth, we use the notation $f(\cdot; \theta): \mathcal{X} \rightarrow \mathcal{Y}$ to denote functions f of measurable domain $\mathcal{X}$ and image space $\mathcal{Y}$ parameterized by some vector $\theta \in \mathbb{R}^d$ with $d \in \mathbb{N}$. We omit the parameters θ when it is clear from the context. We call the set of RL algorithms that make use of parameterized functions to learn a representation of the input states or model the agent's policy as *deep reinforcement learning* (deep RL, or DRL for short). A notable example of parameterized function approximators is *neural networks* (e.g., Schmidhuber, 2015). For completeness, we define the *feed-forward network*, the most common type of neural network used as a nonlinear function approximator.

Definition 15: Feedforward network

A (fully connected) *feed-forward network* is a parameterized function $f(\cdot; \theta): \mathbb{R}^n \rightarrow \mathbb{R}^m$ which consists of a sequence of L layers, each layer $l \in \{1, \dots, L-1\}$ being a function $f_l: \mathbb{R}^{n_{l-1}} \rightarrow \mathbb{R}^{n_l}$ with *weight matrix* $\mathbf{W}^l \in \mathbb{R}^{n_{l-1} \times n_l}$ and a *bias vector* $\mathbf{b}^l \in \mathbb{R}^{n_l}$. The first layer ($l = 0$) is the input layer, and the last layer ($l = L-1$) is the output layer. In between, we call them the *hidden layers*. Each layer l is associated with $n_l \geq 1$ nodes (called *cells* or *neurons*) with nonlinear *activation function* $\sigma_l: \mathbb{R} \rightarrow \mathbb{R}$ (e.g., the sigmoid function $\sigma(x) = 1/(1+\exp(-x))$ or $\text{ReLU}(x) = \max\{0, x\}$), where $n_0 = n$ and $n_{L-1} = m$. Precisely, for $l \in \{1, \dots, L-1\}$, given $\mathbf{z}^{l-1} \in \mathbb{R}^{n_{l-1}}$ and $\mathbf{z}^l \in \mathbb{R}^{n_l}$, we have $f_l(\mathbf{z}^{l-1}) = \mathbf{z}^l$ if and only if $\mathbf{z}_j^l = \sum_{i=1}^{n_{l-1}} \sigma_l(\mathbf{z}_i^{l-1} \cdot \mathbf{W}_{i,j}^l + \mathbf{b}_j^l)$ for all $j \in \{1, \dots, n_l\}$. Then, we have $f(\mathbf{x}; \theta) = f_{L-1} \circ \dots \circ f_1(\mathbf{x})$, with $\mathbf{x} \in \mathbb{R}^n$ and θ so that θ encodes the set of weights and biases $\{\langle \mathbf{W}^l, \mathbf{b}^l \rangle \in \mathbb{R}^{n_{l-1} \times n_l} \times \mathbb{R}^{n_l} \mid 0 < l < L\}$.

The RL algorithms that we present in the sequel will often involve the minimization of *loss functions* L of the form $L(\theta) = \mathbb{E}_{x \sim P} f(x; \theta)$, where $x \in \mathcal{X}$, $P \in \Delta(\mathcal{X})$, for some parameterized function f . Supposing that we are able to sample from P , L can be approximated via *Monte Carlo methods* by drawing a finite number of samples $\{x_i\}_{i=1}^n$ from $\mathcal{X}$ according to P , with $1/n \sum_{i=1}^n f(x_i; \theta) \rightarrow L(\theta)$ as n goes to ∞ , by the strong law of large numbers. The minimization of L according to θ can further be performed in rounds via *stochastic gradient descent* (SGD, Robbins and Monro, 1951):

$$\theta_{t+1} = \theta_t - \alpha_t \cdot \nabla_{\theta_t} f(x_t; \theta_t),$$

where $t \geq 0$ is the *gradient step*, x_t is a sample drawn via P , and α_t is a learning rate, usually assumed to be in the interval $\alpha_t \in (0, 1)$. Alternatively, one can perform a *mini-batch update* at each step, meaning that a batch of samples $\{x_i\}_{i=1}^n$ is drawn at gradient step $t \geq 0$ and the parameters are updated via $\theta_{t+1} = \theta_t - \alpha_t \cdot 1/n \sum_{i=0}^n \nabla_{\theta_t} f(x_i; \theta_t)$. This may lead to smoother convergence in practice. Typical implementations use an *adaptive learning rate* to find a well-suited sequence $(\alpha_t)_{t \in \mathbb{N}}$ according to the specific optimization instance,

possibly depending on the parameters used to design the function f , the loss function L chosen, as well as the current parameters θ_t at each gradient step $t \geq 0$. Popular examples of such adaptive gradient algorithms include RMSPROP [Tieleman and Hinton, 2012] or ADAM [Kingma and Ba, 2015]. Notably, when α_t is appropriately decayed as t goes to ∞ , SGD almost surely converges to a local minimum, and even to the global minimum if L is convex [Robbins and Siegmund, 1971].

2.1 Episodic Processes

In the RL literature, algorithms are broadly categorized into two families: those where the agent is designed for *continual* tasks and those for *episodic* tasks (e.g., Sutton and Barto, 2018).

In the former, the agent continuously interacts within the environment without interruption, so the goal is typically to learn a policy π which maximizes the *average reward* (or *mean-payoff*), defined as $\liminf_{T \rightarrow \infty} 1/T \sum_{t=0}^T \mathbb{E}_{\pi} [\mathcal{R}(s_t, a_t)]$, i.e., the reward obtained in average at the limit of the interaction. In that case, the stationarity of the system under the learned policy is often assumed [Sutton and Barto, 2018; Huang, 2020].

In the latter, the agent trains in *epochs* (referred to as episodes), each culminating in one of three possible outcomes: either the agent (i) achieves success (accomplishing its objective, e.g., reaching a goal state), (ii) encounters failure (visiting an illegal state), or (iii) undergoes termination, indicating the agent is stuck or a time-out has occurred (e.g., Brockman et al., 2016; Pardo et al., 2018). Then, the environment is *reset*.

In this section, we mainly focus on the latter category, as it encapsulates predominant training scenarios in the literature. Formally, an episodic process is defined as follows.

Definition 16: Episodic Process

An MDP $\mathcal{M}$ is *episodic* if and only if its state space $\mathcal{S}$ embeds a special *reset state* $s_{reset} \neq s_I$ so that:

- (i) under any policy π for $\mathcal{M}$, the environment is almost surely eventually reset from every state $s \in \mathcal{S}$: $\mathbb{P}_{\pi}(\Diamond s_{reset} \mid s_I = s) = 1$, where $\Diamond s_{reset} = \{s_{0:\infty}, a_{0:\infty} \mid \text{there exists a } t \geq 0 \text{ so that } s_t = s_{reset}\}$;
- (ii) when reset, the environment restarts from the initial state: for all $a \in \mathcal{A}$,

$$\mathbf{P}(s_I \mid s_{reset}, a) = 1; \text{ and}$$

- (iii) the reset state yields no reward: for all $a \in \mathcal{A}$, $\mathcal{R}(s_{reset}, a) = 0$.

Definition 17: Episode

An *episode* is a finite trajectory $\langle s_{0:T}, a_{0:T-1} \rangle \in \text{Traj}_{\mathcal{M}}^{\text{fin}}$ so that $s_T = s_{\text{reset}}$ and $s_t \neq s_{\text{reset}}$ for all $t < T$. The set of all episodes of $\mathcal{M}$ is denoted as $\text{Episodes}_{\mathcal{M}}$.

Remark 3: Alternative definitions of episodic processes

We note that our definition of episodic MDPs differs from the line of work of Dann and Brunskill [2015]; Dann et al. [2017]; Azar et al. [2017]; Zanette and Brunskill [2019]; Domingues et al. [2021], where episodes *surely* (in contrast to *almost surely*) terminate. Then, there is a fixed horizon $H \geq 0$ which is assumed to be part of the input, i.e., directly included in the instance of any episodic MDP. Notably, in such a setting, *Markovian policies that record the current time step* $t \leq H$ of the interaction *are necessary* to optimize the expected return, which is not the case with our definition.

We rather generally assume that the horizon of every episode is almost surely finite. The aforementioned framework can be reformulated by encoding the current time step t as part of the state space of the model, which is reset when the horizon is reached.

An agent operating within an episodic process seeks to optimize the *episodic return*, whose definition slightly differs from the usual expected return defined in general MDPs (compare with Definition 13):

Definition 18: Episodic Return

Let $\mathcal{M}$ be an episodic process and π be a policy for $\mathcal{M}$, the episodic return of $\mathcal{M}$ is the expected discounted rewards accumulated upon the end of the episode^a:

$$\mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \left(\prod_{i=1}^t \gamma \cdot \mathbb{1}_{\{s_i \neq s_{\text{reset}}\}} \right) \cdot \mathcal{R}(s_t, a_t) \right].$$

^ahere, we assume that $\prod_{i=1}^{t=0} \gamma = 1$.

The related *episodic value functions* are given by

$$Q_{\text{epis}}^{\pi}(s, a) = \begin{cases} \mathcal{R}(s, a) + \mathbb{E}_{s' \sim \mathcal{P}(\cdot|s, a)} [\gamma \cdot V_{\text{epis}}^{\pi}(s')] & \text{if } s \neq s_{\text{reset}}, \text{ and} \\ 0 & \text{otherwise;} \end{cases}$$

and $V_{\text{epis}}^{\pi}(s) = \mathbb{E}_{a \sim \pi(\cdot|s)} [Q_{\text{epis}}^{\pi}(s, a)]$, for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$.

Now, define the *terminal MDP* $\mathcal{M}^\circ$ as the episodic process $\mathcal{M}$ where we solely change the transition function $\mathbf{P}$ so that s_{reset} is *absorbing* (often referred to as the *terminal state* in the RL literature). Precisely, let $\mathbf{P}^\circ$ be the transition function of $\mathcal{M}^\circ$, then $\mathbf{P}^\circ(\cdot | s, a) = \mathbf{P}(\cdot | s, a)$ if $s \neq s_{reset}$ and $\mathbf{P}^\circ(\{s_{reset}\} | s, a) = 1$ otherwise, for any $s \in \mathcal{S}$, $a \in \mathcal{A}$. The episodic return obtained in $\mathcal{M}$ actually corresponds to the usual expected return obtained in $\mathcal{M}^\circ$, which means that

$$\begin{aligned} V_{\text{epis}}^\pi(s) &= \mathbb{E}_\pi^{\mathcal{M}} \left[\sum_{t=0}^{\infty} \left(\prod_{i=1}^t \gamma \cdot \mathbb{1}\{s_i \neq s_{reset}\} \right) \cdot \mathcal{R}(s_t, a_t) \mid s_I = s \right] \\ &= \mathbb{E}_\pi^{\mathcal{M}} \left[\sum_{T=0}^{\infty} \mathbb{1}\{ \langle s_{0:T}, a_{0:T-1} \rangle \in \text{Episodes}_{\mathcal{M}} \} \cdot \sum_{t=0}^T \gamma^t \cdot \mathcal{R}(s_t, a_t) \mid s_I = s \right] \\ &= \mathbb{E}_\pi^{\mathcal{M}^\circ} \left[\sum_{t=0}^{\infty} \gamma^t \cdot \mathcal{R}(s_t, a_t) \mid s_I = s \right]. \end{aligned}$$

Concretely, Q_{epis}^π and V_{epis}^π are thus the usual value functions of the MDP $\mathcal{M}^\circ$. This means that the episodic value functions benefit from exactly the same arsenal of guarantees as those presented in Section 1.5 for the usual value functions. In particular, the episodic Bellman operator $\mathcal{B}_{\text{epis}} : \mathcal{V} \rightarrow \mathcal{V}$ defined as

$$\begin{aligned} (\mathcal{B}_{\text{epis}} V)(s) &= \begin{cases} \sup_{a \in \mathcal{A}} \mathcal{R}(s, a) + \mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, a)} [\gamma \cdot V(s')] & \text{if } s \neq s_{reset}, \text{ and} \\ 0 & \text{otherwise} \end{cases} \\ &= \sup_{a \in \mathcal{A}} \mathcal{R}(s, a) + \mathbb{E}_{s' \sim \mathbf{P}^\circ(\cdot | s, a)} [\gamma \cdot V(s')] \end{aligned}$$

is guaranteed to be an isotone contraction mapping for any $V \in \mathcal{V}$ and $s \in \mathcal{S}$. The same observation remains for the operator applied on the state-action value space, where

$$\begin{aligned} (\mathcal{B}_{\text{epis}} Q)(s, a) &= \begin{cases} \mathcal{R}(s, a) + \mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, a)} [\gamma \cdot \sup_{a'} Q(s', a')] & \text{if } s \neq s_{reset}, \text{ and} \\ 0 & \text{otherwise} \end{cases} \\ &= \mathcal{R}(s, a) + \mathbb{E}_{s' \sim \mathbf{P}^\circ(\cdot | s, a)} \left[\gamma \cdot \sup_{a'} Q(s', a') \right] \end{aligned}$$

for any $Q \in \mathcal{V}_{\mathcal{A}}$, $s \in \mathcal{S}$, and $a \in \mathcal{A}$.

Remark 4: Undiscounted episodic process

In contrast to general MDPs, optimal deterministic stationary policies are guaranteed to exist for the undiscounted ($\gamma = 1$) episodic return in any episodic

MDP with *finite state-action space*, and the solution of the Bellman's equation

$$V_{\text{epis}}^*(s) = \begin{cases} \sup_{a \in \mathcal{A}} [\mathcal{R}(s, a) + \mathbb{E}_{s' \sim P(\cdot|s, a)} [V_{\text{epis}}^*(s')]] & \text{if } s \neq s_{\text{reset}}, \text{ and} \\ 0 & \text{otherwise} \end{cases}$$

is unique [Bertsekas and Tsitsiklis, 1991]. For general state-action spaces, the uniqueness is not necessarily guaranteed [Puterman, 1994, Chapter 7]. Intuitively, the related Bellman operator $\mathcal{B}_{\text{epis}}$, although monotone, *is not a contraction mapping* in infinite state spaces when $\gamma = 1$. Nevertheless, any stationary policy π still does satisfy the Bellman's equation

$$V_{\text{epis}}^\pi(s) = \mathcal{R}_\pi(s) + \mathbb{E}_{s' \sim P_\pi(\cdot|s)} [V_{\text{epis}}^\pi(s')] \cdot \mathbb{1}\{s \neq s_{\text{reset}}\}.$$

Now, given any episodic process, a critical question is whether finding a policy π that optimizes V^π is sufficient and equivalent to finding a policy π' that optimizes $V_{\text{epis}}^{\pi'}$ (e.g., with $\pi' := \pi$). This would mean that optimizing the usual expected return is sufficient to optimize the episodic return. Unfortunately, the answer is negative, as stated in the following lemma.

Lemma 4: Episodic return mismatch ★

In general, a policy maximizing the expected return in an episodic MDP does not necessarily optimize its episodic return. Formally, there exists some episodic MDP $\mathcal{M}$ with state and action spaces $\mathcal{S}$ and $\mathcal{A}$ where

$$\left\{ a \in \mathcal{A} : a = \arg \max_{a' \in \mathcal{A}} Q^*(s, a') \right\} \cap \left\{ a \in \mathcal{A} : a = \arg \max_{a' \in \mathcal{A}} Q_{\text{epis}}^*(s, a') \right\} = \emptyset$$

for some state $s \in \mathcal{S}$.

Proof. Intuitively, the mismatch is a consequence of the discounting effect: even if a policy leads to the optimal episodic return, the horizon required to reach the reset state might yield a low subsequent discount and further lead to a low expected return for the remaining trajectory suffix. In contrast, a policy that maximizes the (usual) expected return may quickly reach the reset while still providing a non-negligible return for the subsequent suffix of the trajectory. Let us construct such an MDP.

Consider the episodic process $\mathcal{M}$ of Figure 2.5. Fix π_a and π_b as the deterministic stationary policies which respectively choose a and b in s_I . Then, we have

$$\begin{aligned} V_{\text{epis}}^{\pi_a}(s_I) &= 2r, \text{ and} \\ V_{\text{epis}}^{\pi_b}(s_I) &= r \cdot (1 + \gamma + \gamma^2). \end{aligned}$$

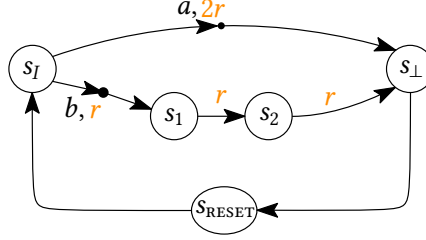


Figure 2.5: Counter-example showing that the optimal episodic and usual expected returns might lead to different policies. The reward $r > 0$ is strictly positive.

The policy π_b is the sole optimal policy for the episodic return of $\mathcal{M}$ whenever $V_{\text{epis}}^{\pi_a}(s_I) < V_{\text{epis}}^{\pi_b}(s_I)$, i.e., when $\gamma > \frac{-1+\sqrt{5}}{2}$. Consider now the usual discounted return. We have

$$V^{\pi_a}(s_I) = 2r + \gamma \cdot V^{\pi_a}(s_{\perp}) = 2r + \gamma^2 \cdot V^{\pi_a}(s_{\text{reset}}) = 2r + \gamma^3 \cdot V^{\pi_a}(s_I)$$

and

$$\begin{aligned} V^{\pi_b}(s_I) &= r + \gamma V^{\pi_b}(s_1) = r + \gamma \cdot r + \gamma \cdot V^{\pi_b}(s_2) = r + \gamma \cdot r + \gamma^2 r + \gamma^3 \cdot V^{\pi_b}(s_{\perp}) \\ &= r \cdot (1 + \gamma + \gamma^2) + \gamma^3 \cdot V^{\pi_b}(s_{\perp}) \\ &= r \cdot (1 + \gamma + \gamma^2) + \gamma^4 \cdot V^{\pi_b}(s_{\text{reset}}) \\ &= r \cdot (1 + \gamma + \gamma^2) + \gamma^5 \cdot V^{\pi_b}(s_I). \end{aligned}$$

Then,

$$\begin{aligned} V^{\pi_a}(s_I) &= 2r + \gamma^3 \cdot V^{\pi_a}(s_I) & V^{\pi_b}(s_I) &= r \cdot (1 + \gamma + \gamma^2) + \gamma^5 \cdot V^{\pi_b}(s_I) \\ &= \frac{2r}{1 - \gamma^3} & &= \frac{r \cdot (1 + \gamma + \gamma^2)}{1 - \gamma^5}, \end{aligned}$$

Observe that $V^{\pi_a}(s_I) > V^{\pi_b}(s_I)$ whatever the discount γ . Then, in the particular case where $\gamma > \frac{-1+\sqrt{5}}{2}$, we have that π_b and π_a are respectively the optimal policy for the episodic return and the usual expected return of $\mathcal{M}$ and

$$\arg \max_{a^* \in \{a, b\}} Q^*(s_I, a^*) = a \neq b = \arg \max_{a^* \in \{a, b\}} Q_{\text{epis}}^*(s_I, a^*).$$

□

For the sake of presentation, in the remainder of this chapter, when referring to the value functions of episodic processes, we rather consider their episodic return. That is, except

if otherwise specified, for every episodic process, we override the notation $V := V_{\text{epis}}$, $Q := Q_{\text{epis}}$, and $\mathcal{B} := \mathcal{B}_{\text{epis}}$.

Remark 5: About the stationarity of terminal MDPs.

In the following, we show that any episodic process $\mathcal{M}$ is (or can be made without loss) ergodic, which implies the existence of a unique stationary distribution ξ_π under any stationary policy π for $\mathcal{M}$. When using stationary measures to compute a quantity written as the expectation of a random variable (e.g., possibly involving V_{epis}), it is worth noting that *we do consider the distribution defined over the original episodic MDP $\mathcal{M}$, and not over the terminal MDP $\mathcal{M}^\circ$ where the stationary distribution is trivially defined as the Dirac function with impulse on state s_{reset} .*

Ergodicity. Interpreting any episodic process as an MDP that is almost surely infinitely often reset (see Definition 16) introduces a perspective that diverges from their traditional treatment in the RL literature. Typically, RL algorithms handle the continual case — where the agent is optimized across an infinite MDP rollout — and the episodic case — where finite trajectories are drawn to optimize the finite-horizon return — as distinct scenarios [Sutton and Barto, 2018]. Our approach aligns with the perspective of Huang [2020], where learning across episodes is viewed as an infinite unrolling of the MDP, akin to the continual case. We demonstrate in the following that the underlying training environment is ergodic (in both the finite and general cases), ensuring the existence of a stationary distribution in all episodic scenarios (cf. Section 1.4 for a detailed discussion about the ergodicity of MDPs). This implies that sampling transitions from the episodic process under a particular policy can be conceptualized as drawing transitions collected along a single rollout. Specifically, every point in the trajectory obtained by running a stationary policy for a sufficient number of time steps (up to convergence) is guaranteed to be an unbiased sample of the induced stationary distribution [Lovász and Winkler, 1995; Propp and Wilson, 1998; Huang, 2020].

Throughout this thesis, adopting this perspective enables us to formally reason about and analyze the underlying learning procedures of RL algorithms and further derive corresponding guarantees.

Theorem 7: Episodic processes are ergodic $\star$

Proof. Let $\mathcal{M}$ be an episodic process and π be a stationary policy for $\mathcal{M}$. We first show that $\mathcal{M}_\pi$ is irreducible in the finite case and Harris recurrent in the general case. Intuitively, this result follows from the fact that every state reachable from s_I allows to almost surely eventually visit s_{reset} , which transitions to s_I . So all states are visited infinitely often with probability one. More concretely:

The finite case. Assume $\mathcal{M}$ has a finite state space and fix π in $\mathcal{M}$. We say that a state s is *connected* to another state s' , written $s \rightsquigarrow s'$, if and only if $\xi_\pi^T(s' | s) > 0$ for some $T > 0$. Notice that the relation $\rightsquigarrow$ is transitive by definition. We show that the set of states reachable under π from s_I , written $B = \{s \mid s_I \rightsquigarrow s\} \subseteq \mathcal{S}$, forms a unique BSCC since we have $\mathbb{P}_\pi(\Diamond s_{reset} \mid s_I = s) = 1$: it is clear that all states $s \in B$ are connected to s_I since s almost surely reaches s_{reset} and $\mathbf{P}_\pi(s_I \mid s_{reset}) = 1$ by Definition 16. Then, $s \rightsquigarrow s_{reset} \rightsquigarrow s_I$. Consequently, we have for all pairs $s, s' \in B$ that $s \rightsquigarrow s'$ since $s \rightsquigarrow s_I$ and $s_I \rightsquigarrow s'$. So, B is strongly connected (cf. Definition 9). Furthermore, B is the largest subset of $\mathcal{S}$ containing s_I : by definition, all states outside B are not reachable from s_I . Then B is an SCC.

It remains to show that B is a *bottom* SCC, i.e., B satisfies $\mathbf{P}_\pi(B \mid s) = 1$ for any $s \in B$. Assume this is not the case, i.e., $\mathbf{P}_\pi(B \mid s) < 1$ for some $s \in B$. This would mean that there is some state $s' \in \mathcal{S} \setminus B$ so that $s \rightsquigarrow s'$. This is not compliant with the definition of B since this means that $s_I \rightsquigarrow s \rightsquigarrow s'$, and by transitivity that $s_I \rightsquigarrow s'$. Thus s' needs to be part of B since it is assumed to embed all states that are connected to s_I .

Then, we have that B is a BSCC, so the MC $\mathcal{M}_\pi$ is irreducible.

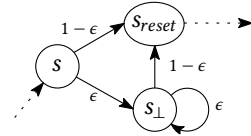
The general case. We show that $\mathcal{M}_\pi$ is a recurrent Harris chain (cf. Definition 11). The proof is closely related to Example 9, where we show that a Markov chain with restart is Harris recurrent. Let $H = \{s_{reset}\}$. By Definition 16, it is clear that H is recurrent. Furthermore, H is also a small set: let $\rho \in \Delta(\mathcal{S})$ be the Dirac measure defined as $\rho(C) = \mathbb{1}_{\{s_I \in C\}}$, then we have $\mathbf{P}_\pi(\cdot \mid s_{reset}) = \rho$.

Now, it remains to show that the process is aperiodic. While it is not always the case (e.g., in Figure 2.5, the MCs induced by deterministic stationary policies are irreducible but not aperiodic), one can enforce aperiodicity without any impact on the episodic value functions. Intuitively, one can enforce the aperiodicity of the system by injecting a small amount of noise into the transition probabilities. This modification is referred to as the *recursive ϵ -perturbation trick* by Huang [2020]. The idea is to add a dummy state $s_\perp$ to $\mathcal{S}$ so that all states that are supposed to transition to s_{reset} first transition to $s_\perp$ before with a low probability $\epsilon > 0$. Formally, we modify $\mathbf{P}$ to $\mathbf{P}'$ so that

$$\mathbf{P}'(s_{reset} \mid s, a) = (1 - \epsilon) \cdot \mathbf{P}(s_{reset} \mid s, a)$$

and $\mathbf{P}'(s_\perp \mid s, a) = \epsilon$ for all state-action pairs $\langle s, a \rangle$ in

$$\{s, a \in \mathcal{S} \times \mathcal{A} \mid \mathbf{P}(s_{reset} \mid s, a) > 0\} \cup \{s_\perp\}.$$



We also require that the rewards yielded by $s_\perp$ are null: $\mathcal{R}(s_\perp, a) = 0$ for all $a \in \mathcal{A}$. Notice that the latter condition ensures the fact that V_{epis} and Q_{epis} undergo no change as a result of this perturbation.

Then, the aperiodicity is enforced: intuitively, for any episode of length T , there exists another episode of length $T + 1$ with non-zero probability which involves the dummy state $s_\perp$ at the end of the episode, just before reaching s_{reset} . Precisely, take any episode $\langle s_{0:T}, a_{0:T-1} \rangle \in \text{Episodes}_{\mathcal{M}}$, then for all $T' > T$, there exists another episode $\langle s'_{0:T'}, a'_{0:T'-1} \rangle \in \text{Episodes}_{\mathcal{M}}$ where $s_t = s'_t$, $a_t = a'_t$, and $s'_i = s_\perp$ for all $0 \leq t < T \leq i < T'$. Then, $\xi_{\pi}^{T'}(s_{\text{reset}}) > 0$.

Since (a) $\mathcal{M}_\pi$ is either irreducible or Harris recurrent under any stationary policy π , and (b) one can enforce the aperiodicity of the system without impact on the episodic value functions, we conclude by (a) and (b) that $\mathcal{M}$ is ergodic. $\square$

Corollary 1: Episodic processes converge to the stationary distribution

Let $\mathcal{M}$ be an episodic process and π be a stationary policy of $\mathcal{M}_\pi$. Then, $\mathcal{M}_\pi$ has a unique stationary distribution ξ_π . Furthermore,

- (i) if $\mathcal{M}_\pi$ is aperiodic, $\lim_{t \rightarrow \infty} \xi_\pi^t = \xi_\pi$ is the unique stationary policy of $\mathcal{M}_\pi$;
- (ii) otherwise, if $\mathcal{M}_\pi$ is periodic, slightly perturb the process via the ϵ -perturbation trick described in the proof of Theorem 7, let $\mathcal{M}_\pi^\epsilon$ be the resulting MC for some $\epsilon \in (0, 1)$ and $\xi_\pi^{t,\epsilon}$ be the transient distribution of $\mathcal{M}_\pi^\epsilon$ for any $t \geq 0$. Then, the unique stationary measure of $\mathcal{M}_\pi^\epsilon$ is given by $\lim_{t \rightarrow \infty} \xi_\pi^{t,\epsilon} = \xi_\pi^\epsilon$ and $\xi_\pi \propto \xi_\pi^\epsilon$, with

$$\xi_\pi(S) = \frac{\xi_\pi^\epsilon(S)}{1 - \xi_\pi^\epsilon(s_\perp)}$$

for all $S \in \Sigma(S)$ [Huang, 2020].

In practice, this corollary means that when one needs to evaluate or approximate the expectation of a random variable over the stationary distribution, one simply has to perturb the environment *if necessary*, execute the policy within the (possibly perturbed) environment, and let it converge to the stationary distribution. When a sufficient number of time steps have elapsed, we obtain an unbiased sampling point. If the process has been perturbed, one just needs to rescale the quantity to $1/(1 - \xi_\pi^\epsilon(s_\perp))$.

Alternatively, one can obtain a (biased) Monte Carlo estimation of such an expectation by uniformly sampling batches of transitions occurring along a trajectory: when the number of time steps elapsed $t \geq 0$ becomes very large, this is close to sampling from the occupancy measure μ_π^t , which almost surely converges to ξ_π . Enlarging t reduces the bias of the estimation; this intuitively allows to de-correlate transitions randomly sampled along the trajectory.

Remark 6: Existence of a unique stationary distribution

We stress once again that while a unique stationary distribution is typically *assumed* in continual RL scenarios [Sutton and Barto, 2018], it is *guaranteed to exist* in the episodic setting. Consequently, in the formal analysis of RL algorithms we will present in this dissertation, we often assume the existence of such a stationary distribution.

Effects of the ϵ -perturbation trick. Note that the ϵ -perturbation trick presented in the proof of Theorem 7 further allows to control the speed of convergence of any episodic process to the stationary distribution: the time steps that the agent spend in the dummy state $s_\perp$ follows the geometric distribution with expectation $\epsilon/1-\epsilon$. Interestingly, Huang [2020] empirically found that setting ϵ to $1 - 1/\mathbb{E}_\pi[T]$, where T is a random variable capturing the length of every episode, tends to enforce the convergence to the stationary distribution within $T = 3 \mathbb{E}_\pi[T]$ steps in various environments.

In the remainder of this section, we assume that the environment is episodic, and we give an overview of value-based and actor-critic methods. While the former can be seen as a counterpart of VI in the RL setting, the latter are generalized PI algorithms suited for learning through interactions within an unknown environment.

2.2 Value-based Methods

Q-learning

One of the most popular vanilla RL algorithms is *Q-learning* [Watkins and Dayan, 1992], a *stochastic form of value iteration* which is designed for finite MDPs. Intuitively, Q-learning maintains a table taking the form of a vector $\theta \in \mathbb{R}^{|S \times \mathcal{A}|}$ (the so-called *Q-table*) which allows learning the optimal Q-value of each state-action pair. The Q-value function is usually merely implemented as a look-up function to the Q-table θ : S and $\mathcal{A}$ being finite, one can enumerate all state-action pairs from $S \times \mathcal{A}$ as $\langle s^{(i)}, a^{(i)} \rangle$ for all $1 \leq i \leq |S \times \mathcal{A}|$, with $Q(s^{(i)}, a^{(i)}; \theta)$ being defined as the i^{th} entry of θ .

The Q-values are learned iteratively *off-policy*. At each step $t \geq 0$ of the interaction, a transition $\langle s_t, a_t, r_t, s_{t+1} \rangle$ is drawn in $\mathcal{M}$, and the Q-table θ_t is updated to θ_{t+1} according to the following equation: for all $s \in S$, $a \in \mathcal{A}$,

$$Q(s, a; \theta_{t+1}) = Q(s, a; \theta_t) + \alpha_t \left(r_t + \gamma \max_{a' \in \mathcal{A}} Q(s_{t+1}, a'; \theta_t) - Q(s, a; \theta_t) \right) \quad (2.15)$$

if $s = s_t \neq s_{\text{reset}}$ and $a = a_t$; and $Q(s, a; \theta_{t+1}) = Q(s, a; \theta_t) \cdot \mathbb{1}\{s \neq s_{\text{reset}}\}$ otherwise, for a given learning rate $\alpha_t \in (0, 1)$. The policy of the agent is further defined as $\pi(a | s; \theta) =$

$\mathbb{1} \{a = \arg \max_{a' \in \mathcal{A}} Q(s, a'; \theta)\}$ for all states $s \in \mathcal{S}$. Then, the **update** rule of Figure 2.4 consists in updating θ_t to θ_{t+1} by applying Equation 2.15.

Exploration. In practice, π^b can be seen as a Markovian policy. In the particular context of Q -learning, $(\pi_t^b)_{t \in \mathbb{N}}$ is usually implemented as an ϵ -greedy policy. Under such a policy, the agent acts greedily with probability $(1 - \epsilon_t)$ and randomly with probability ϵ_t :

$$\pi_t^b(a | s) = \begin{cases} (1 - \epsilon_t) & \text{if } a = \arg \max_{a' \in \mathcal{A}} Q(s, a'; \theta_t); \text{ and} \\ \epsilon_t / |\mathcal{A}| & \text{otherwise,} \end{cases} \quad (2.16)$$

for any $s \in \mathcal{S}$, $a \in \mathcal{A}$, and $0 < \epsilon_t < 1$. A good practice is to further decay ϵ_t to zero along the training process, as t goes to ∞ . Another popular choice is the use of *Boltzmann policies*. Intuitively, the $\arg \max$ operator over the actions maximizing the Q -values that defines the deterministic distribution of the agent's policy is replaced by a *softmax*:

$$\pi_t^b(a | s) = \frac{\exp(Q(s, a; \theta_t) / \lambda_t)}{\sum_{a' \in \mathcal{A}} \exp(Q(s, a'; \theta_t) / \lambda_t)} \quad (2.17)$$

for a given temperature parameter $0 < \lambda_t < 1$. Again, annealing the temperature λ_t to zero as t goes to ∞ is a good practice and allows to eventually retrieve $\mathbb{1} \{a = \arg \max_{a' \in \mathcal{A}} Q(s, a'; \theta_t)\}$ at the limit of the interaction.

Guarantees. Notably, when the learning rate α_t of Equation 2.15 is decayed appropriately⁷ and when π^b ensures to almost surely visit all state-action pairs infinitely often along training steps $t \geq 0$, then $Q(\cdot; \theta_t)$ is guaranteed to almost surely converge to Q^* as t goes to ∞ [Watkins and Dayan, 1992; Jaakkola et al., 1993; Tsitsiklis, 1994].

Q -learning can also be modified to provide *probably approximately correct* (PAC) guarantees. Intuitively, given an error $\epsilon > 0$ and confidence $\delta > 0$, the RL algorithm is PAC if one can find a lower bound $T \geq 0$ on the number of interaction steps required to ensure that for all $t \geq T$, the policy π at time step t is ϵ -optimal with probability at least $1 - \delta$. To provide such guarantees, the update rule of Equation 2.15 can be modified to *delay* the updates of θ : the Q -value of each state-action pair is updated only when the pair is experienced a sufficient number of times, and the update is then averaged over the recently missed updates opportunities [Strehl et al., 2006].

Beyond the tabular setting, convergence guarantees can be preserved for general state spaces. Nowé [1994] proved that the convergence to the optimal values holds by using piecewise linear interpolation of the Q -function, and Tsitsiklis and Roy [1997] further proved the result using more general *linear function approximators* to model the Q -function. However, the use of general *nonlinear function approximators* to model the

⁷Precisely, $(\alpha_t)_{t \in \mathbb{N}}$ should satisfy the so-called *Robbins-Monro condition* [Robbins and Monro, 1951].

Q -function breaks all guarantees linked to the optimality of the algorithm in general state spaces.

Deep Q -learning

Deep Q -networks (DQN) [Mnih et al., 2015] maintain an approximation of the Q -values by modeling the Q -function through (deep) *neural networks* (the so-called Q -networks), which are parameterized nonlinear function approximators. Convergence guarantees are here sacrificed in favor of scalability to general state space environments, while the action space is still assumed finite.

Precisely, as vanilla Q -learning, DQN seeks Q^* , i.e., the optimal (Q -)value function. Recall that $\max_{a \in \mathcal{A}} Q^*(s, a)$ is the *unique* solution of the Bellman's Equation 2.10, where

$$Q^*(s, a) = (\mathcal{B}Q^*)(s, a) = \mathcal{R}(s, a) + \mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, a)} \left[\gamma \cdot \max_{a' \in \mathcal{A}} Q^*(s', a') \right].$$

Assume that π^b is Markovian, let $Q(\cdot; \theta) \in \mathcal{V}_{\mathcal{A}}$ be the Q -network parameterized by weights θ , and let the random variable

$$Y(s, a, r, s') = r + \gamma \cdot \max_{a' \in \mathcal{A}} Q(s', a'; \hat{\theta}) \quad (2.18)$$

denote the *target value* obtained for the transition $\langle s, a, r, s' \rangle$. Such transitions are drawn from the interaction of $\mathcal{M}$ when the agent executes π^b . Precisely, after $t \geq 0$ time steps, we consider transition drawn from the occupancy measure $\mu_{\pi^b}^t$ (cf. Equation 2.2).

Importantly, given $s \in \mathcal{S}, a \in \mathcal{A}$, and $r = \mathcal{R}(s, a)$, $(\mathcal{B}Q(\cdot; \hat{\theta}))(s, a) = \mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, a)} [Y]$. Notice also that we generally consider that Y embeds its own set of parameters $\hat{\theta}$. A DQN agent tries adjusting the parameters θ of the Q -network $Q(\cdot; \theta)$ by minimizing the *mean-squared Bellman error* (MSBE) between $Q(s, a; \theta)$ and the expected target Y , to eventually retrieve the unique fixed point of the Bellman's equation so that $Q(\cdot; \theta)$ almost surely matches $\mathcal{B}Q(\cdot; \hat{\theta})$ by *minimizing* the following loss to zero:⁸

$$\begin{aligned} L_{\text{MSBE}}(\theta) &= \mathbb{E}_{s, a, r \sim \mu_{\pi^b}^t} \left[\left(\mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, a)} [Y] - Q(s, a; \theta) \right)^2 \right] \\ &= \mathbb{E}_{s, a, r \sim \mu_{\pi^b}^t} \left[\mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, a)} [Y]^2 + Q(s, a; \theta)^2 - 2 \mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, a)} [Y] Q(s, a; \theta) \right] \end{aligned} \quad (2.19)$$

⁸Note that our derivation of the MSBE leads to subtracting the marginal variance $\text{Var}_Y = \mathbb{E}_{s, a \sim \mu_{\pi^b}^t} [\mathbb{V}_{s' \sim \mathbf{P}(\cdot | s, a)} [Y]]$ from $L_{\text{DQN}}(\theta) = \mathbb{E}_{s, a, r, s' \sim \mu_{\pi^b}^t} [(Y - Q(s, a; \theta))^2]$ in Equation 2.20, whereas Mnih et al. [2015] curiously derives **adding** Var_Y to $L_{\text{DQN}}(\theta)$. Our analysis agrees with the one of Fan et al. [2020].

$$\begin{aligned}
&= \mathbb{E}_{s,a,r,s' \sim \mu_{\pi^b}^t} [Q(s,a;\theta)^2 - 2YQ(s,a;\theta)] + \mathbb{E}_{s,a,r \sim \mu_{\pi^b}^t} \left[\mathbb{E}_{s' \sim \mathbf{P}(\cdot|s,a)} [Y]^2 \right] \\
&= \mathbb{E}_{s,a,r,s' \sim \mu_{\pi^b}^t} [Q(s,a;\theta)^2 - 2YQ(s,a;\theta) + Y^2] + \\
&\quad \mathbb{E}_{s,a,r \sim \mu_{\pi^b}^t} \left[\mathbb{E}_{s' \sim \mathbf{P}(\cdot|s,a)} [Y]^2 - \mathbb{E}_{s' \sim \mathbf{P}(\cdot|s,a)} [Y^2] \right] \\
&= \mathbb{E}_{s,a,r,s' \sim \mu_{\pi^b}^t} [(Y - Q(s,a;\theta))^2] - \mathbb{E}_{s,a \sim \mu_{\pi^b}^t} \left[\mathbb{V}_{s' \sim \mathbf{P}(\cdot|s,a)} [Y] \right]. \tag{2.20}
\end{aligned}$$

Interestingly, when θ and $\hat{\theta}$ are distinct, the marginal variance $\mathbb{E}_{s,a \sim \mu_{\pi^b}^t} [\mathbb{V}_{s' \sim \mathbf{P}(\cdot|s,a)} [Y]]$ is independent from θ , so differentiating $L_{\text{MSBE}}(\theta)$ with respect to θ yields the following gradients:

$$\nabla_{\theta} L_{\text{MSBE}}(\theta) = -2 \mathbb{E}_{s,a,r,s' \sim \mu_{\pi^b}^t} \left[\left(r + \gamma \cdot \max_{a' \in \mathcal{A}} Q(s', a'; \hat{\theta}) - Q(s,a;\theta) \right) \cdot \nabla_{\theta} Q(s,a;\theta) \right]. \tag{2.21}$$

Furthermore, if $\hat{\theta}$ is periodically updated to θ after T_{target} steps and if $\hat{\theta}$ is fixed during this period, the optimization of L_{MSBE} w.r.t. θ (Equation 2.19) coincides with the minimization of

$$\mathbb{E}_{s,a,r \sim \mu_{\pi^b}^t} \left[\left(\left(\mathcal{B}Q(\cdot; \hat{\theta}) \right)(s,a) - Q(s,a;\theta) \right)^2 \right].$$

The optimization procedure taking place during this period of T_{target} steps can then be seen as an approximate⁹ stochastic VI step (Algorithm 1), where the MSBE is minimized instead of the usual Bellman error.

Since the agent has no access to $\mathbf{P}$ and the gradient of L_{MSBE} coincides with those of $\mathbb{E}_{s,a,r,s' \sim \mu_{\pi^b}^t} [(Y - Q(s,a;\theta))^2]$ *when separate target parameters are used*, it is sufficient for a DQN agent to merely minimize the loss function

$$L_{\text{DQN}}(\theta) = \mathbb{E}_{s,a,r,s' \sim \mu_{\pi^b}^t} \left[\left(r + \gamma \cdot \max_{a' \in \mathcal{A}} Q(s', a'; \hat{\theta}) - Q(s,a;\theta) \right)^2 \right]. \tag{2.22}$$

Furthermore using SGD to minimize L_{DQN} w.r.t. θ , the gradient descent step of the parameters θ is given by

$$\theta_{t+1} = \theta_t + \alpha_t \cdot \left(r + \gamma \cdot \max_{a' \in \mathcal{A}} Q(s', a'; \hat{\theta}_t) - Q(s,a;\theta_t) \right) \cdot \nabla_{\theta_t} Q(s,a;\theta_t), \tag{2.23}$$

⁹or exact if (i) the global minimum is reached during the procedure and (ii) the Q -network is powerful enough to model $\mathcal{B}Q(\cdot; \hat{\theta})$, i.e., if there exist some weights θ' so that θ' minimizes L_{MSBE} (cf. Equation 2.19), where $Q(\cdot; \theta')$ almost surely matches $\mathcal{B}Q(\cdot; \hat{\theta})$, w.r.t. $\mu_{\pi^b}^t$.

where $\alpha_t \in (0, 1)$ is a learning rate for $t \geq 0$ and $\langle s, a, r, s' \rangle$ is drawn from $\mu_{\pi^b}^t$. Equation 2.23 can be seen as the gradient counterpart of the Q -learning update rule (Equation 2.15).

To efficiently practically learn the Q -network by minimizing $L_{\text{DQN}}(\theta)$, two tricks introduced by Mnih et al. [2015] are crucial.

Experience replay. The transitions collected during the interaction of the agent executing π^b in $\mathcal{M}$ are sequentially correlated, and thus yield biased samples of $\mu_{\pi^b}^t$ for $t \geq 0$ interactions steps. An *experience replay buffer* [Lin, 1992] can be introduced to alleviate this issue, that is used to *emulate* $\mu_{\pi^b}^t$ by storing transitions encountered along the interaction. Intuitively, at each step $t \geq 0$ of the interaction, the agent *stores* the transition $\langle s_t, a_t, r_t, s_{t+1} \rangle$ in $\mathcal{D}$. Then, a *batch* $\left\{ \langle s_i, a_i, r_i, s'_i \rangle \right\}_{i=1}^N$ of N samples is drawn from $\mathcal{D}$ uniformly¹⁰, and $L_{\text{DQN}}(\theta)$ is approximated through $1/N \sum_{i=1}^N (y_i - Q(s_i, a_i; \theta))^2$ where $y_i = r_i + \gamma \max_{a' \in \mathcal{A}} Q(s'_i, a'; \hat{\theta})$ is the target.

In practice, the buffer size of $\mathcal{D}$ is usually very large (Mnih et al., 2015 use a size of 10^6), so sampling from $\mathcal{D}$ is close to sampling independent transitions from $\mu_{\pi^b}^t$. Drawing transitions randomly from $\mathcal{D}$ allows breaking the strong sequential correlation of transitions collected during the interaction, further reducing the variance of the updates.

Target network. As mentioned above, the gradients derived in Equation 2.21 are sound if and only if a *target network* $Q(\cdot; \hat{\theta})$ is used with its own distinct parameters $\hat{\theta}$. On the contrary, when $\theta = \hat{\theta}$, minimizing L_{DQN} is significantly different from minimizing the MSBE since the minimization of L_{DQN} involves minimizing the marginal variance of the target as well (cf. Equation 2.20). Additionally, an optimization step would push the parameters of both the Q - and target networks at the same time in the gradient direction, which further usually leads to oscillations and the divergence of the policy. Thus, a target network with separate parameters allows for stabilizing the overall learning procedure.

In practice, $\hat{\theta}$ is periodically updated to match θ at regular intervals: given a *target update rate* $\beta \in (0, 1]$, the target update rule is given by $\hat{\theta}' = \beta \cdot \theta + (1 - \beta) \cdot \hat{\theta}$. When $\beta = 1$, the rule is a *hard target update*, as used by Mnih et al. [2015], while $\beta < 1$ leads to a *soft target update* which corresponds to applying *Polyak averaging* [Polyak and Juditsky, 1992] to the target parameters (e.g., as used by Haarnoja et al., 2018).

Learning the policy. The gradient step implemented through the **update** function of the RL loop (Figure 2.4) is detailed in Algorithm 3. Note that the DQN policy can be implemented as a pure greedy policy with regards to the Q -network learned. However,

¹⁰Note that the subscript here corresponds to the i^{th} transition sampled from $\mathcal{D}$, and not the i^{th} transition visited along the interaction at time step $t = i$.

Algorithm 3: DQN template**Input:**

- Respective parameters $\theta, \hat{\theta}$ of the Q - and the target networks;
- Current interaction step $t \geq 0$;
- Replay buffer $\mathcal{D}$;
- $(\alpha_t)_{t \in \mathbb{N}}, \beta$: Q -network learning and target network update rates;
- $T_{\text{learn}}, T_{\text{target}}$ Q -network learning and target update periods with $T_{\text{learn}} \leq T_{\text{target}}$;
- $P : \mathcal{S} \times \mathcal{V}_{\mathcal{A}} \rightarrow \Delta(\mathcal{A})$ policy distribution template (e.g., ϵ -greedy or Boltzmann);

function `update`($\pi(\cdot; \theta), s, a, r, s'$)

$\mathcal{D} \leftarrow \mathcal{D} \cup \{(s, a, r, s')\}$ $\triangleright$ store the transition $\langle s, a, r, s' \rangle$ in $\mathcal{D}$

if $t \bmod T_{\text{learn}} = 0$ **then**

$\{\langle s_i, a_i, r_i, s'_i \rangle\}_{i=1}^N \sim \mathcal{D}$ $\triangleright$ sample a batch of transitions from $\mathcal{D}$

forall $i \leftarrow 1$ to N **do**

$y_i \leftarrow r_i + \max_{a' \in \mathcal{A}} Q(s'_i, a'; \hat{\theta}) \cdot \mathbb{1}\{s_i \neq s_{\text{reset}}\}$ $\triangleright$ compute the target

$\triangleright$ Perform a (mini-)batched gradient step

$$\theta \leftarrow \theta + \alpha_t \cdot \frac{1}{N} \sum_{i=1}^N (y_i - Q(s_i, a_i; \theta)) \cdot \nabla_{\theta} Q(s_i, a_i; \theta)$$

if $t \bmod T_{\text{target}} = 0$ **then**

$\hat{\theta} \leftarrow \beta \cdot \theta + (1 - \beta) \cdot \hat{\theta}$ $\triangleright$ smooth target update

return π so that $\pi(a | s; \theta) = P(a | s, Q(\cdot; \theta))$

since we have no guarantee on the optimality of the related Q -function, injecting a small amount of randomness (e.g., via an ϵ -greedy policy) is a good practice to prevent the agent from being blocked by an action believed to be optimal (Mnih et al., 2015 evaluate the agent using $\epsilon = 0.05$). Alternatively, one can implement the final policy as a Boltzmann policy to exploit the *meta-information* learned through the Q -networks. Indeed, while the outputted Q -function has no optimality guarantee, $Q(s, a; \theta)$ tells us how confident the learning agent is on the optimality of the action $a \in \mathcal{A}$ in every state $s \in \mathcal{S}$, so using a Boltzmann policy to turn this information into a distribution may be better suited in states whose approximated Q values are close.

Further tips and tricks. The extensions of DQN are numerous and include the addition of a second Q -network to prevent the DQN agent from overestimating the values of the state-action pairs [van Hasselt et al., 2016], the replacement of $\mathcal{D}$ by a *prioritized*

experience replay buffer to replay important transitions more frequently and learn more efficiently [Schaul et al., 2016], or a modification of the Q -function in a distributional perspective [Bellemare et al., 2023]. All of these extensions are compiled into a cohesive algorithm called *Rainbow DQN* [Hessel et al., 2018]. This algorithm not only outperforms other DQN variants in experimental performance but also demonstrates state-of-the-art performance within the deep RL landscape.

Guarantees. We note that when sparse neural networks solely composed of rectified linear units (ReLU, Nair and Hinton, 2010) are employed, DQN generates a sequence of policies whose corresponding value functions converge to the optimal values when both the sample size and the number of iterations tend to infinity [Fan et al., 2020]. However, achieving those guarantees is extremely challenging and is significantly constrained by practical limitations. First, one needs to have access to a *fixed* distribution over the state-action space in replacement of the replay buffer. Second, each optimization period needs to reach the global minimum of the least mean square Bellman error, while the application of the Bellman operator on the Q -network must remain *accurately modelable* by the neural network itself (essentially the conditions discussed in footnote 9, where the capability of the Q -network to model $\mathcal{B}Q$ is relaxed). Precisely, the application of the Bellman operator to the Q -network should yield a composition of Hölder smooth functions; this guarantees the Q -network to approximately be capable of modeling $\mathcal{B}Q$.

2.3 Actor-critic Methods

Generalized policy iteration (GPI) embeds all algorithms implementing PI (Algorithm 2) so that the evaluation step **eval** only partially evaluates the current policy. Indeed, the *global* evaluation (i.e., over the whole state-action space) may quickly become intractable, in particular when considering general state-action spaces. GPI is often implemented through the *actor-critic* paradigm: intuitively, at each optimization step, the *critic evaluates* the values of the current policy, and the *actor improves* the policy based on the critic evaluation.

Note that due to the partial evaluation, *GPI does not generally leverage the policy improvement guarantee* provided by Lemma 3. Then, the policies learned may oscillate during their training if a local optimum is found, even if the evaluation step is exact [Bertsekas and Tsitsiklis, 1996]. Nevertheless, GPI embeds methods compliant with general state-action spaces, and guarantees can be derived from some well-designed algorithms, relying on *Mirror descent* theory [Kuba et al., 2022].

The Actor: policy improvement. We first consider the policy improvement step. Let π be a stationary policy for $\mathcal{M}$ with parameters θ . The improvement step of θ performed

by the actor involves the maximization of the discounted return in the terminal¹¹ MC $\mathcal{M}_\pi^\circ$:

$$J(\theta) = \mathbb{E}_\pi^{\mathcal{M}^\circ} \left[\sum_{t=0}^{\infty} \gamma^t \cdot \mathcal{R}(s_t, a_t) \right]. \quad (2.24)$$

In order to modify the parameters θ to further increase $J(\theta)$ through stochastic gradient methods, we define the *discounted* occupancy measure¹² $\mu_\pi^\gamma \in \Delta(\mathcal{S})$ of each state in the terminal process $\mathcal{M}_\pi^\circ$ for the discount factor γ as the (normalized) discounted probability of visiting that state along the interaction:

$$\mu_\pi^\gamma(\mathcal{S}) = (1 - \gamma) \cdot \sum_{t=0}^{\infty} \gamma^t \mathbb{P}_\pi^{\mathcal{M}^\circ}(\{s_0:\infty, a_0:\infty \mid s_t \in \mathcal{S}\}). \quad (2.25)$$

One can see μ_π^γ as a discounted counterpart of the stationary distribution $\xi_\pi = \lim_{t \rightarrow \infty} \mu_\pi^t$ in the original MDP $\mathcal{M}_\pi$. Intuitively, $\mu_\pi^\gamma(\mathcal{S})$ describes the expected proportion of time the agent spends in the region $\mathcal{S} \subseteq \mathcal{S}$ when executing π , which is weighted by the discount factor γ . Importantly, the expected return from the initial state can be reformulated using this measure [Puterman, 1994]:

$$J(\theta) = \mathbb{E}_\pi^{\mathcal{M}^\circ} \left[\sum_{t=0}^{\infty} \gamma^t \cdot \mathcal{R}(s, a) \right] = \frac{1}{1 - \gamma} \cdot \mathbb{E}_{s, a \sim \mu_\pi^\gamma} \mathcal{R}(s, a), \quad (2.26)$$

where we use the same notation $s, a \sim \mu_\pi^\gamma$ as for ξ_π , i.e., first sampling s from μ_π^γ , then a from $\pi(\cdot \mid s; \theta)$. Importantly, notice that since the occupancy measure μ_π^γ gives a distribution over states visited when $\pi(\cdot; \theta)$ is executed, the gradients of $J(\theta)$ w.r.t. θ *does depend on* $\mu_{\pi(\cdot; \theta)}^\gamma$, which is hardly tractable. Interestingly, Sutton et al. [1999a] proved that the gradients of $J(\theta)$ can be rewritten as an analytical expression that is independent of the gradients of μ_π^γ .

Theorem 8: Policy gradient

For any episodic MDP $\mathcal{M}$, let π be a stationary policy for $\mathcal{M}$ with parameters θ ,

(i) [Sutton et al., 1999a]: for $\gamma < 1$, the gradient of $J(\theta)$ is given by

$$\nabla_\theta J(\theta) = \frac{1}{1 - \gamma} \cdot \mathbb{E}_{s \sim \mu_\pi^\gamma} \left[\int_{\mathcal{A}} Q^\pi(s, a) \cdot \nabla_\theta \pi(\mathrm{d}a \mid s; \theta) \right];$$

¹¹the following discussion (including Theorem 8) also holds in the general, non-episodic setting. However, in that case, we only allow for $\gamma < 1$. One just needs to replace $\mathcal{M}^\circ$ by $\mathcal{M}$ in the equations.

¹² μ_π^γ is defined here for $\mathcal{M}^\circ$, but it can be trivially lifted to $\mathcal{M}$ (see footnote 11).

Theorem 8: Policy gradient (*continued*)

(ii) [Huang, 2020]: assuming the objective is undiscounted ($\gamma = 1$), let ξ_π be the stationary distribution of $\mathcal{M}_\pi$, then

$$\nabla_\theta J(\theta) = (\mathbb{E}_\pi [\mathbf{T}] - 1) \cdot \mathbb{E}_{s \sim \xi_\pi} \left[\int_{\mathcal{A}} Q^\pi(s, a) \cdot \nabla_\theta \pi(\mathrm{d}a \mid s; \theta) \mid s \neq s_{\text{reset}} \right],$$

where $\mathbf{T} : \text{Traj}_\mathcal{M} \rightarrow \mathbb{N}$, $\langle s_0 : \infty, a_0 : \infty \rangle \mapsto \sum_{T=0}^{\infty} (T+1) \mathbb{1} \{ s_0 : T, a_0 : T-1 \in \text{Episodes}_\mathcal{M} \}$ is the *length of the (first) episode* of the trajectory, while $1/\xi_\pi(s_{\text{reset}}) = \mathbb{E}_\pi [\mathbf{T}]$ is the *average episode length (AEL)* [Serfozo, 2009, Chapter 1, Theorem 54].

Remark 7: Average episode length and adaptive learning rate

In the undiscounted case, Theorem 8 slightly differs from the classic policy gradient theorem introduced by Sutton et al. [1999a], where the AEL term is omitted and treated as a proportionally constant. We contend that the AEL *does depend* on $\pi(\cdot; \theta)$ and is thus not a constant: changing the policy changes the AEL as well. While omitting the term does not alter the gradient direction, it does affect the learning rate control [Huang, 2020].

Interestingly, for the state distribution $d_\pi \in \Delta(S)$ so that

$$d_\pi = \begin{cases} \mu_\pi^\gamma & \text{if } \gamma < 1, \text{ and} \\ \xi_\pi(\cdot \mid \{s \in S \mid s \neq s_{\text{reset}}\}) & \text{otherwise,} \end{cases}$$

Assuming π has density p_π , one can rewrite the integration over the whole action space as a tractable expectation:

$$\begin{aligned} \nabla_\theta J(\theta) &\propto \mathbb{E}_{s \sim d_\pi} \left[\int_{\mathcal{A}} Q^\pi(s, a) \cdot \nabla_\theta \pi(\mathrm{d}a \mid s; \theta) \right] \\ &= \mathbb{E}_{s \sim d_\pi} \left[\int_{\mathcal{A}} p_\pi(a \mid s; \theta) \cdot Q^\pi(s, a) \cdot \frac{\nabla_\theta p_\pi(a \mid s; \theta)}{p_\pi(a \mid s; \theta)} \mathrm{d}a \right] \\ &= \mathbb{E}_{s \sim d_\pi} \left[\int_{\mathcal{A}} \pi(\mathrm{d}a \mid s; \theta) \cdot Q^\pi(s, a) \cdot \nabla_\theta \log p_\pi(a \mid s; \theta) \right] \\ &\quad \text{(because } \frac{\partial}{\partial \theta} \log p_\pi(\cdot; \theta) = \frac{1}{p_\pi(\cdot; \theta)} \cdot \frac{\partial}{\partial \theta} p_\pi(\cdot; \theta)) \\ &= \mathbb{E}_{s, a \sim d_\pi} [Q^\pi(s, a) \cdot \nabla_\theta \log p_\pi(a \mid s; \theta)]. \end{aligned}$$

This enables the optimization of $J(\theta)$ through stochastic gradient ascent algorithms.

The Critic: policy evaluation. In practice, $Q^\pi(s, a)$ can be estimated through Monte-Carlo methods, using the reward accumulated along episodes drawn by executing π : for any trajectory τ whose prefix is an episode $\langle s_{0:T}, a_{0:T-1} \rangle \in \text{Episodes}_{\mathcal{M}}$, one can estimate $Q^\pi(s_t, a_t)$ via $G_t^T(\tau) = \sum_{i=t}^{T-1} \gamma^{i-t} r_i$ with $r_t = \mathcal{R}(s_t, a_t)$ for $t < T$. This results in an unbiased estimation of $Q^\pi(s, a)$ when $s = s_t$ and $a = a_t$ since

$$\mathbb{E}_{\pi}^{\mathcal{M}} \left[\sum_{T=0}^{\infty} \mathbb{1} \{ s_{0:T}, a_{0:T-1} \in \text{Episodes}_{\mathcal{M}} \} \cdot G_t^T \mid s_t = s, a_t = a \right] = Q^\pi(s, a).$$

An alternative approach is to introduce a neural network $Q(\cdot; \eta)$ with parameters η whose updates are based on transitions observed during the execution of the current policy π . The optimization of the parameters η may then follow similar update procedures and tricks to those employed in DQN, as used by *soft actor-critic* (SAC) agents [Haarnoja et al., 2018]. Notably, in the context of policy evaluation, the max operator in Equation 2.18 vanishes in favor of an action a' drawn from π , as the focus is on seeking the values of the current policy.

Nevertheless, the variance of policy gradients arising from such estimations tends to be high. A *baseline* $b : \mathcal{S} \rightarrow \mathbb{R}$ is typically subtracted from the (estimated) Q -values to mitigate the variance of policy updates. Notably, the policy gradient theorem still holds in the presence of the baseline:

$$\nabla_{\theta} J(\theta) \propto \mathbb{E}_{s, a \sim d_{\pi}} [(Q^\pi(s, a) - b(s)) \cdot \nabla_{\theta} \log p_{\pi}(a \mid s; \theta)].$$

A typical choice is $b(s) = V^\pi(s)$, resulting in the so-called *advantage* function:

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s).$$

Then, in practice, a value network $V(\cdot; \eta)$ can be learned (instead of $Q(\cdot; \eta)$) to estimate the advantage function. Note that episodes have often different length according to the current policy, so a good practice is to fix a horizon $n \geq 0$ and consider the *n-step return* collected along episodes $\langle s_{0:T}, a_{0:T}, r_{0:T} \rangle$, then *bootstrap* the values of the remaining steps from $V(s_n; \eta)$ to estimate $Q^\pi(s_t, a_t)$:

$$A_t = \begin{cases} G_t^{t+n} + \gamma^n \cdot V(s_n; \eta) - V(s_t; \eta) & \text{if } t + n < T \\ G_t^T - V(s_t; \eta) & \text{otherwise.} \end{cases} \quad (2.27)$$

This is precisely the critic implemented in *Advantage Actor-critic* (A2C) agents [Mnih et al., 2016]. The general **update** procedure of A2C agents is presented in Algorithm 4.

Discussion. In essence, the inherent practical challenges associated with actor-critic algorithms encompass those associated with GPI. Notably, in the interval between each

Algorithm 4: Advantage Actor-critic template

Input:

- Actor and Critic parameters θ, η ;
- Current interaction step $t \geq 0$;
- *Sequentially ordered* replay buffer $\mathcal{D}$;
- Advantage estimation function **advantage** : $\mathbb{R} \times \mathcal{V} \times \text{Traj}_{\mathcal{M}}^{\text{fin}} \rightarrow \mathbb{R}$ (e.g., Equation 2.27 or the one introduced by Schulman et al., 2016);
- Value network $V(\cdot; \eta) \in \mathcal{V}$;
- Actor and Critic learning rates $(\alpha_t)_{t \in \mathbb{N}}, (\beta_t)_{t \in \mathbb{N}}$;
- Learning update period $\mathcal{T}$ and return horizon n ;

function `update`($\pi(\cdot; \theta), s, a, r, s'$)

```

▷ store the transition  $\langle s, a, r, s' \rangle$  occurring at time step  $t$  in  $\mathcal{D}$ 
 $\mathcal{D} \leftarrow \mathcal{D} \cup \{\langle s, a, r, s' \rangle\}$ 
if  $t \bmod \mathcal{T} = 0$  then
  ▷ Critic
  for  $i \leftarrow 0$  to  $\mathcal{T} - 1$  do
     $H \leftarrow \min\{i + n, \mathcal{T}\}$  ▷ horizon for the  $n$ -step return
    ▷ Compute the  $n$ -step return from transitions  $\langle s_i, a_i, r_i, s'_i \rangle$  stored in  $\mathcal{D}$ 
     $G_i \leftarrow \sum_{j=i}^{H-1} [(\prod_{k=0}^j \gamma \cdot \mathbb{1}\{s_k \neq s_{\text{reset}}\}) \cdot r_j]$ 
    ▷ Compute the target return by bootstrapping from  $V(\cdot; \eta)$  if necessary
    if  $s_{\text{reset}} \notin \{s_i, s_{i+1}, \dots, s_H\}$  then
       $y_i \leftarrow G_i + \gamma^n \cdot V(s_H; \eta)$ 
    else
       $y_i \leftarrow G_i$ 
  ▷ Value network update
   $\eta \leftarrow \eta + \beta_t \cdot \frac{1}{\mathcal{T}} \sum_{i=0}^{\mathcal{T}-1} (y_i - V(s_i; \eta)) \cdot \nabla_{\eta} V(s_i; \eta)$ 
  ▷ Policy evaluation
  for  $i \leftarrow 0$  to  $\mathcal{T} - 1$  do
     $A_i \leftarrow \text{advantage}(G_i, V(\cdot; \eta), \mathcal{D})$ 
  ▷ Actor: policy improvement
   $\theta \leftarrow \theta + \alpha_t \cdot \frac{1}{\mathcal{T}} \sum_{i=0}^{\mathcal{T}-1} A_i \cdot \nabla_{\theta} \log p_{\pi}(a_i | s_i; \theta)$ 
  clear  $\mathcal{D}$ 
return  $\pi$ 

```

policy improvement step, the policy evaluation step demands the construction of a suitable estimator for the value function, which is challenging.

Actor-critic methods additionally suffer from learning instabilities [Zhao et al., 2012]; to further stabilize the training procedures, certain algorithms restrict the *drift* that occurs between policy updates within a defined neighborhood, ensuring the theoretical improvement of the policy. Examples of such algorithms include *trust region policy optimization* (TRPO) [Schulman et al., 2015] and *proximal policy optimization* (PPO) [Schulman et al., 2017], both implementing more generalized advantage estimation techniques [Schulman et al., 2016]. Notably, TRPO and PPO are instances of Mirror Learning [Kuba et al., 2022], ensuring convergence to optimal values *when the policy evaluation step is exact and when the states are drawn from the suitable distribution d_π* .

Remark 8: Shambles in ad hoc policy gradient algorithms

A critical problem occurring in most of the practical implementations of actor-critic algorithms is related to this *suitable distribution d_π* . All of the aforementioned state-of-the-art methods use a discount factor $\gamma < 1$, collect data *on-policy*, store them in a buffer $\mathcal{D}$, and update the parameters θ according to the gradient given by

$$\nabla_\theta J_\gamma(\theta) = \mathbb{E}_{s,a \sim \mathcal{D}} [Q^\pi(s,a) \cdot \nabla_\theta \log p_\pi(a \mid s; \theta)].$$

When the number of data collected t is sufficiently large, we are close to sampling from μ_π^t (the *undiscounted* occupancy measure, cf. Section 1.4, Equation 2.2), which almost surely converges to ξ_π as t goes to ∞ . However, as it is wrongly assumed in all those methods, $\nabla_\theta J_\gamma(\theta) \neq \nabla_\theta J(\theta)$ since *the samples used to update the policy are not distributed according to the discounted occupancy measure μ_π^γ but they are rather collected on-policy, i.e., distributed according to the stationary measure ξ_π (as $t \rightarrow \infty$), which intuitively boils down to removing the discounting effect in the occupancy measure^a μ_π^γ* . This further means that all those algorithms do not optimize the correct objective function. Notably, most of the papers related to state-of-the-art policy gradient methods even made erroneous or misleading claims regarding the discounted policy gradient theorem. This important mismatch between theory and practice has been discussed and analyzed by Nota and Thomas [2020], which showed that optimizing the policy through $J_\gamma(\theta)$ even leads to the worst possible policy in some well-chosen environments.

Nevertheless, surprisingly, computing $\nabla_\theta J_\gamma(\theta)$ does not lead to poor performance in practice — on the contrary, state-of-the-art methods implementing this biased version of policy gradient are very effective. This leads to questioning the reason for the effectiveness of those algorithms and whether those performances can be

further improved by optimizing the correct objective function. In particular, it is unclear whether fixing this bias would lead to better performance, or whether the bias is, actually, beneficial in practice.

Fixing the bias. This bias can be fixed by introducing the missing discount factor γ^t in the *dual* formulation of the policy gradient theorem — where full trajectories are considered instead of transitions — since

$$\mathbb{E}_{s \sim \mu_\pi^\gamma} [Q^\pi(s, a) \cdot \nabla_\theta \log p_\pi(a | s; \theta)] = \mathbb{E}_\pi^{\mathcal{M}^\odot} \left[\sum_{t=0}^{\infty} \gamma^t \cdot Q^\pi(s, a) \cdot \nabla_\theta \log p_\pi(a | s; \theta) \right].$$

Equivalently, one can also consider sampling from μ_π^γ , which boils down to sampling from the stationary distribution of the episodic environment where we add a special state s_γ to the state space that is entered with probability $(1 - \gamma)$ at each time step, as discussed by Puterman [1994, Chapter 5.3] and White [2017]. Concretely, this means one needs to modify $\mathbf{P}$ to $\gamma \cdot \mathbf{P}(S | s, a) + (1 - \gamma) \cdot \mathbb{1} \{s_\gamma \in S\}$, for any $s \in S$, $a \in \mathcal{A}$, and $S \in \Sigma(S)$. This obviously poses sample efficiency issues, so one could alternatively take inspiration from off-policy and *importance sampling* techniques to draw from the stationary distribution of the original environment and rescale the gradients according to a learned approximation of the ratio $\frac{\mu_\pi^\gamma}{\xi_\pi}$, similarly to Gelada and Bellemare [2019].

Finally, since the discount is the core of the problem, one could simply question whether discounting is really important and merely use the undiscounted version of the gradient theorem. However, even though undiscounted objectives are possible in the context of episodic RL (cf. Remark 4), this amounts to dealing with the *stochastic longest path problem*^b in the terminal MDP [Bertsekas and Tsitsiklis, 1991], which is a relatively different problem coming with its own set of challenges. In particular, in general state spaces, the Bellman operator is not a contraction mapping and the uniqueness of the solution of the Bellman’s equation is no longer guaranteed.

^aNote that we can even obtain unbiased samples from ξ_π in the episodic setting (cf. Section 2.1).

^bWe note that the *stochastic shortest path problem* studied by Bertsekas and Tsitsiklis [1991] involves the minimization of the expected costs along trajectories, which does *admit negative costs*. The maximization of the episodic return can be merely reduced to the stochastic shortest path to the terminal state by inverting the sign of all rewards, to turn them into costs.

2.4 Are Deep RL Algorithms Reliable?

Throughout this section, we have explored deep RL algorithms that leverage function approximators to model the value function, the policy, or even both, to scale to large or even continuous state-action spaces.

Value-based methods are essentially stochastic VI algorithms. A notable example is Q -learning, which employs a lookup table to maintain the value of each state-action pair and is further equipped with convergence guarantees at the limit of the interaction. However, this approach struggles to scale to general state spaces, necessitating the introduction of function approximators to effectively scale to reasonable settings. Although linear function approximators and piecewise interpolation can preserve some guarantees, they all vanish with the use of general function approximators. Despite the loss of guarantees, employing neural networks to model the Q -value, as seen in the DQN algorithm, has yielded significant performance improvements compared to previous value-based methods. DQN can be modified to additionally provide convergence guarantees, but this modification highly limits the applicability of the algorithm, relying on unrealizable assumptions in practice (a fixed dataset, an exact optimization procedure, and a sparse ReLU network capable of approximately capturing the Bellman operator).

Actor-critic methods on the other hand, are generalized PI algorithms. The policy gradient theorem provides a principled approach to optimize the policy through gradient ascent methods. Policy gradient is further compliant with general nonlinear function approximators and some algorithms can even be connected to Mirror learning, which additionally provides convergence guarantees to an optimal policy.

The gradient computation involves the estimation of the value function, which often leads to high variance. This variance can be corrected by subtracting a baseline from the estimation. A popular solution is learning a value function through a neural network, which is further used for computing the advantage function and bootstrapping the value when a finite horizon is considered. This neural network introduction for learning the value function, however, leads to the same challenges as those discussed in the previous paragraph.

Besides the loss of guarantees incurred by the introduction of a value network, the policy gradient theorem is also often misused in practice, leading to a biased objective function (cf. Remark 8). Nevertheless, this bias does not seem to degrade the performance of the algorithms in practice, and the reason for this still remains unclear.

So, are deep RL algorithms reliable? Following this discussion, it becomes evident that the reliability of deep RL algorithms is an open issue. Specifically, in addition to robust and principled RL algorithms, there is a significant need to delve into understanding the effectiveness of deep RL policies and their resulting behaviors.

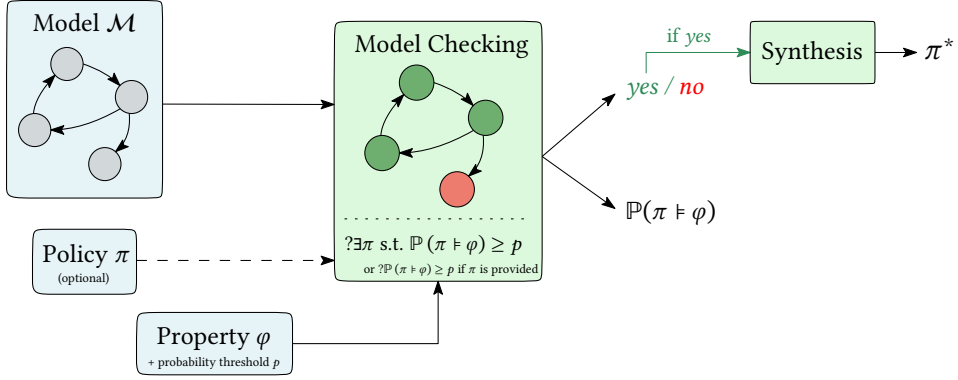


Figure 2.6: Model checking and synthesis of stochastic systems.

In this dissertation, we endeavor to address these questions to some extent. We adopt an approach relying on formal methods to comprehensively grasp and analyze the behaviors exhibited by an agent when operating under *any* deep RL policies generated by *any* algorithm. This involves formally modeling the interaction between RL policies and the environment, employing a technique known as *model checking*, to formally verify that the policies adhere to desired behavioral properties. Further details on this formal verification technique are introduced in the following section.

3 Model Checking and Synthesis

In the previous section, we assumed that the environment is unknown, i.e., that the agent is able to interact within the environment and observe the consequence of its actions, but without having explicit access to the model prescribing its underlying dynamics. We also described algorithms compliant with general state-action spaces. However, we have seen that the techniques used to scale to large problems lead to the loss of any useful guarantee, further impacting the reliability of the underlying algorithms.

Let us take a step backward and assume on the contrary that the model of the environment is known (i.e., explicitly given) and finite. Such a setting is well suited to formally analyze and reason about the behaviors of the agent. In particular, in this section, we focus on *model checking* [Baier and Katoen, 2008], a model-based formal method that consists of formally verifying whether a set of properties, expressed in some logic, holds in a given model.

In the context of stochastic model checking, the model of the environment is often considered as being an MDP $\mathcal{M}$. The properties we consider in this thesis are expressed

in a temporal logic, such as *linear temporal logic* (LTL), which allows to specify properties such as *safety* (something bad never happens), *reachability* (eventually accessing some region of the state space), *liveness* (something good keeps happening), or *persistence* (something good eventually happens and keeps happening). More generally, such logics allow to express ω -regular properties over infinite traces of the model.

The high-level idea of model checking is depicted in Figure 2.6. The model $\mathcal{M}$, the property φ , and a probability threshold p are given as input of the model checking algorithm, along with a policy π for $\mathcal{M}$ if one requires to verify the specific behaviors of the agent when π is executed in $\mathcal{M}$. The model checker generally performs an exhaustive exploration of the state space of $\mathcal{M}$ to formally verify whether there exists a policy that allows satisfying the property φ with at least (or at most) probability p (or whether π satisfies φ with probability p if provided). The model checker then outputs “yes” or “no”, depending on the result of the verification, along with the maximum (resp. minimum) probability of satisfying φ . Finally, if the answer is “yes”, *synthesis* is a step further which consists in constructing a policy π^* that allows satisfying φ with at least (resp. at most) probability p .

The first part of this section is devoted to the formalization of the stochastic model checking problem and is sketched as follows. We first introduce the so-called *linear temporal logic* (LTL). We show that the latter allows expressing a rich subset of all the properties over the measurable fragment of the (infinite) trajectory space of the MDP (i.e., properties that can be expressed through countable union, intersection, and complementation of cylinders). Then, we focus on the particular reachability property, whose model checking procedure consists of verifying whether a region of the system is reachable with a sufficient probability. Intuitively, this property constitutes the cornerstone for the verification of all other properties. We further show that any LTL formula can be translated into a *deterministic Rabin automaton* (DRA), a deterministic finite automaton that allows for generating words of infinite length. By doing the product of the MDP and the DRA representing the LTL formula, we obtain a new MDP in which the model checking problem can be reduced to the reachability to particular regions of the resulting product. Finally, we discuss the challenges and limitations of model checking properties in the context of reinforcement learning.

3.1 Linear Temporal Logic

We start by defining the syntax and semantics of LTL, to further express specification through this logical formalism.

Definition 19: Syntax of LTL

The syntax of LTL formulae over the set **AP** is defined by the following grammar:

$$\varphi := \mathbf{a} \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \circ\varphi \mid \varphi_1 \mathbf{U} \varphi_2$$

where $\mathbf{a} \in \mathbf{AP}$ is an atomic proposition, $\neg$ is the *negation operator*, $\wedge$ is the *conjunction operator*, $\circ$ is the *next operator*, and $\mathbf{U}$ is the *until operator*.

The *size* of a formula φ , denoted $|\varphi|$, is the number of recursive calls needed to parse φ through the grammar.

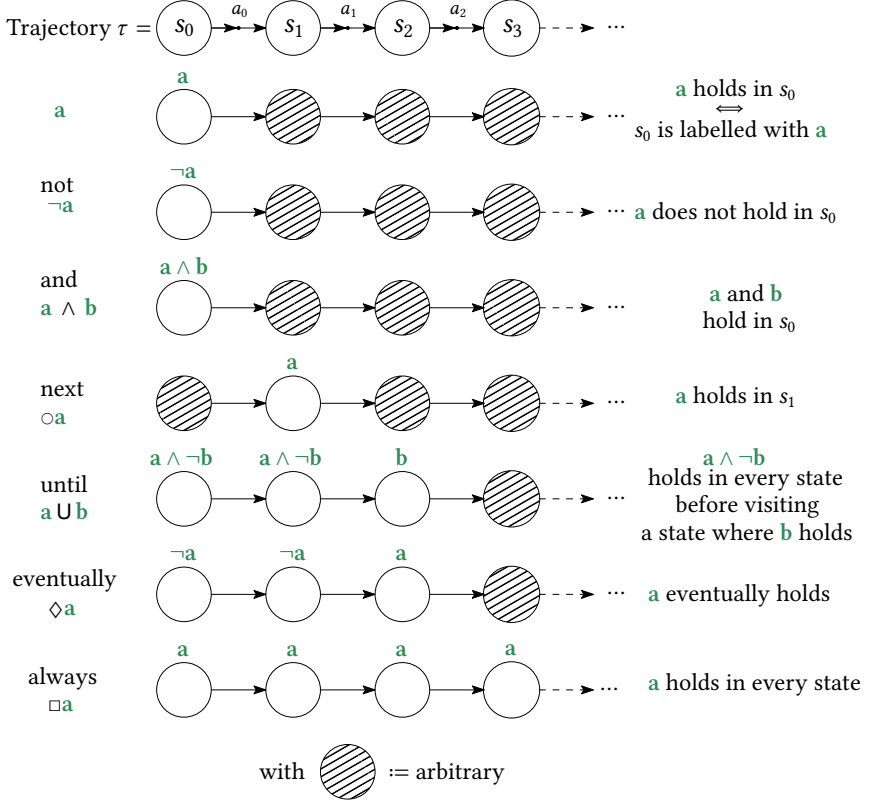
Intuitively, $\circ\varphi$ means that the formula φ holds at the next time step, while $\varphi_1 \mathbf{U} \varphi_2$ means that φ_1 holds at the current step and will hold during the next steps until φ_2 holds. LTL allows for standard logical connective abbreviations such as $\varphi_1 \vee \varphi_2 \equiv (\neg\varphi_1 \wedge \neg\varphi_2)$. The true proposition is further defined as $\text{true} \equiv \mathbf{a} \vee \neg\mathbf{a}$, for any $\mathbf{a} \in \mathbf{AP}$.

Temporal operators can further be derived: $\Diamond\varphi \equiv \text{true} \mathbf{U} \varphi$ means that φ will *eventually* hold along the trajectory, and $\Box\varphi \equiv \neg\Diamond\neg\varphi$ means that φ *always* hold, at any time step.

Figure 2.7 depicts the intuitive meaning of LTL operators in the simple case where the formulae consist of atomic propositions. Intuitively, in an MDP, we say that an atomic proposition $\mathbf{a}$ holds at a time step t if s_t is labeled with $\mathbf{a}$, i.e., $\mathbf{a} \in \ell(s_t)$. Also notice in Definition 19 that the syntax permits the combination of temporal operators. As concrete examples, the formula $\Diamond\Box\varphi$ means that φ will eventually hold and will keep holding forever, whereas $\Box\Diamond\varphi$ means that φ holds infinitely often.

Semantics. The semantics of LTL formulae are defined over the space of infinite trajectories of the MDP $\mathcal{M}$. We say that a trajectory $\tau = \langle s_{0:\infty}, a_{0:\infty} \rangle$ satisfies a formula φ , denoted $\tau \models \varphi$, when the following conditions hold:

$\tau \models \mathbf{a}$	if and only if	$\mathbf{a} \in \ell(s_0)$
$\tau \models \neg\varphi$	if and only if	$\tau \not\models \varphi$
$\tau \models \varphi_1 \wedge \varphi_2$	if and only if	$\tau \models \varphi_1$ and $\tau \models \varphi_2$
$\tau \models \circ\varphi$	if and only if	$\langle s_{1:\infty}, a_{1:\infty} \rangle \models \varphi$
$\tau \models \varphi_1 \mathbf{U} \varphi_2$	if and only if	there exists a step $i \geq 0$ from which $\langle s_{i:\infty}, a_{i:\infty} \rangle \models \varphi_2$ and for all $0 \leq j < i$, $\langle s_{j:\infty}, a_{j:\infty} \rangle \models \varphi_1$.

Figure 2.7: Intuitive semantics of LTL operators, for $a, b \in \text{AP}$.**Example 11: LTL properties in the grid world environment**

Consider the MDP of our running example (Example 3), that we slightly modify to embed a passage behind the adversary , as depicted in Figure 2.8. Properties of interest in this example include the following LTL formulae:

- $\Box \neg \text{devil}$: the agent should always avoid the adversary ;
- $\Diamond \Box \checkmark$: the agent should eventually remain in a safe region of the environment;
- $\Diamond \text{target}$: the agent should eventually reach the target ;
- $\neg \text{target} \cup (\text{cherry} \wedge \Diamond \text{target})$: the agent should at least eat a cherry before going to the target; and

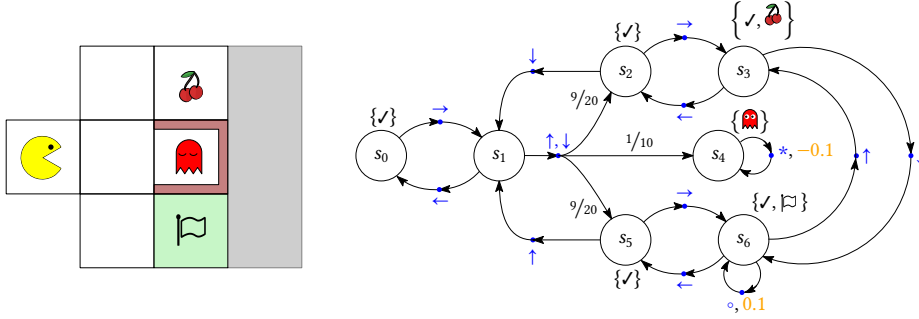


Figure 2.8: Back to our running example (Example 3). The environment is slightly modified and now embeds a passage between the cell labeled with and the target cell . This passage allows the agent to sneak up behind the opponent without the latter waking up.

- $\Box(\Diamond \text{flag} \wedge \Diamond \text{cherry})$: the agent should continuously reach the target and eat cherries.

The model checking problem in an MDP $\mathcal{M}$ consists of verifying whether there exists a policy whose induced trajectories allow satisfying a given LTL formula φ with a predefined minimum (resp. maximum) probability. If the policy is provided, then the model checker outputs the probability with which the policy satisfies φ .

First of all, we focus on reachability properties, which intuitively constitute the cornerstone for the verification of all other properties.

3.2 Reachability

Reachability properties have the form $\Diamond \varphi$, where φ is an LTL formula. The simplest formula consists of a single atomic proposition $\varphi = a$, which means that the agent must eventually visit a state labeled with a . This event is measurable since it can be expressed as the countable union or intersection of cylinder sets where

$$\{\tau \in \text{Traj}_{\mathcal{M}} \mid \tau \models \Diamond a\} = \bigcup_{t \geq 0} \bigcup_{\tau \in \text{Pref}_{\mathcal{M}}^t} \text{Cyl}(\tau),$$

where $\text{Pref}_{\mathcal{M}}^t = \{s_{0:t}, a_{0:t-1} \in \text{Traj}_{\mathcal{M}}^{\text{fin}} \mid a \in \ell(s_t)\}$ is the set of finite trajectories of length t whose last state is labeled with a . In fact, this event is not only measurable in the finite setting, but also in the context involving general state spaces (cf. Equation 2.8).

Therefore, verifying $\mathcal{M}$ for the reachability property $\Diamond a$ and the lower probability bound p (resp. upper bound p) consists in computing $\sup_{\pi} \mathbb{P}_{\pi}(\{\tau \in \text{Traj}_{\mathcal{M}} \mid \tau \models \Diamond a\})$

(resp. the infimum of the same quantity), and then checking if the result agrees with the given threshold.

In keeping with tradition, for any initial state, this quantity is a solution of Bellman's equation modified to take into account the reachability property.

Theorem 9: Bellman's equation for reachability [Forejt et al., 2011]

Let $\mathbf{a} \in \mathbf{AP}$ be an atomic proposition and $\text{opt} \in \{\max, \min\}$ be the optimization direction. Recall that the transient distribution $\xi_\pi^t(E \mid s)$ gives the probability of being in a state from $E \subseteq \mathcal{S}$ in exactly t steps from s (see Equation 2.1 in Section 1.4).

Then, define $\mathcal{S}_{=1} = \{s \in \mathcal{S} \mid \mathbf{a} \in \ell(s)\}$ as the set of states labeled with $\mathbf{a}$, and $\mathcal{S}_{=0}$ so that $s \in \mathcal{S}_{=0}$ if and only if

- (i) there is no stationary policy π nor $t \geq 0$ so that $\xi_\pi^t(\mathcal{S}_{=1} \mid s) > 0$ whenever $\text{opt} = \max$, and
- (ii) there exists a stationary policy π for $\mathcal{M}$ so that $\xi_\pi^t(\mathcal{S}_{=1} \mid s) = 0$ for any $t \geq 0$, whenever $\text{opt} = \min$.

Then, $\mathcal{S}_{=0}$ is the set of states that are disconnected from $\mathcal{S}_{=1}$ under *all* policies when $\text{opt} = \max$ (resp. under *some* policy when $\text{opt} = \min$). Finally, define $\mathcal{S}_? = \mathcal{S} \setminus (\mathcal{S}_{=1} \cup \mathcal{S}_{=0})$. The optimal value function $V_{\Diamond \mathbf{a}}^*$ for the reachability property $\Diamond \mathbf{a}$ is a solution of the following Bellman's equation:

$$V_{\Diamond \mathbf{a}}^*(s) = \begin{cases} \text{opt}_{a \in \mathcal{A}} \mathbb{E}_{s' \sim \mathbf{P}(\cdot \mid s, a)} V_{\Diamond \mathbf{a}}^*(s') & \text{for all } s \in \mathcal{S}_?, \\ 1 & \text{for all } s \in \mathcal{S}_{=1}, \text{ and} \\ 0 & \text{for all } s \in \mathcal{S}_{=0}. \end{cases} \quad (2.28)$$

Furthermore,

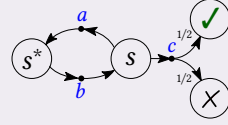
- (i) $V_{\Diamond \mathbf{a}}^*(s) = \sup_\pi \mathbb{P}_\pi(\{\tau \in \text{Traj}_{\mathcal{M}} \mid \tau \models \Diamond \mathbf{a}\} \mid s_I = s)$ for all $s \in \mathcal{S}$ is the *least solution* of Equation 2.28 when $\text{opt} = \max$, whereas
- (ii) $V_{\Diamond \mathbf{a}}^*(s) = \inf_\pi \mathbb{P}_\pi(\{\tau \in \text{Traj}_{\mathcal{M}} \mid \tau \models \Diamond \mathbf{a}\} \mid s_I = s)$ for all $s \in \mathcal{S}$ is the *unique solution* of Equation 2.28, when $\text{opt} = \min$.

Note that $\mathcal{S}_{=0}$ can be computed in polynomial time via purely graph algorithms, merely by checking the connectivity of states to $\mathcal{S}_{=1}$. As usual, we write $V_{\Diamond \mathbf{a}}^\pi$ for the value function of the MC $\mathcal{M}_\pi$ induced by a policy π for $\mathcal{M}$ in Equation 2.28 (where the opt operator over the action space vanishes since the latter consists of a singleton).

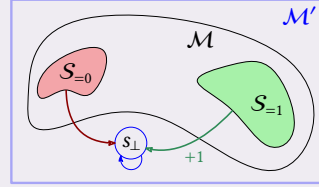
Remark 9: Uniqueness of the solution

Notice that in the case where $\text{opt} = \max$, Bellman's equation is not guaranteed to have a unique solution.

For instance, consider the MDP depicted in the right figure. Consider the reachability property $\Diamond\checkmark$. The value function $V := \{s^* \mapsto 1, s \mapsto 1, \checkmark \mapsto 1, X \mapsto 0\}$ is a solution of Equation 2.28 for $\text{opt} = \max$. However, it clearly does not correspond to the values of the reachability property $\Diamond\checkmark$, i.e., $V_{\Diamond\checkmark}^* := \{s^* \mapsto 1/2, s \mapsto 1/2, \checkmark \mapsto 1, X \mapsto 0\}$.

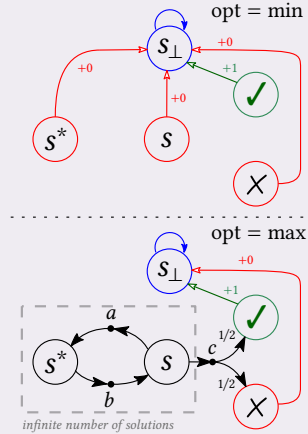


To understand why the solution is not necessarily unique, construct the MDP $\mathcal{M}'$ from any MDP $\mathcal{M}$ so that $\gamma = 1$ and where states from $S_{=1}$ as well as those from $S_{=0}$ transition deterministically to a new absorbing state $s_\perp$, whatever the action chosen. We set the rewards of $\mathcal{M}'$ to zero everywhere, except when a state from $S_{=1}$ transitions to $s_\perp$, in which case a reward of one is earned.



Note that the Bellman's equation for the (undiscounted) expected return in $\mathcal{M}'$ (Equation 2.10) coincides with the one for the reachability property in $\mathcal{M}$ (Equation 2.28).^a Since the expected return from any states in $\mathcal{M}'$ is comprised in $[0, 1]$, it is indeed guaranteed to be a solution of the Bellman's equation.

However, since the return is undiscounted, the solution is generally not unique [Puterman, 1994, Chapter 7.1.2], and the Bellman operator $\mathcal{B}$ is not guaranteed to be a contraction mapping. In the particular case where $\text{opt} = \min$, all states from which at least one policy does not ensure to almost surely reach $S_{=1}$ in $\mathcal{M}$ are embedded into $S_{=0}$. Therefore, in contrast to the maximization case, $s_\perp$ is almost surely reachable from every state under any stationary policy, either through states from $S_{=1}$ or $S_{=0}$. Since $s_\perp$ is absorbing, the Bellman operator $\mathcal{B}$ applied on $\{V \in \mathcal{V} \mid V(s_\perp) = 0\}$ is a contraction mapping and the Bellman's equation has a unique solution [Bertsekas and Tsitsiklis, 1991]. On the other hand, when $\text{opt} = \max$, the maximum reachability probabilities can be expressed as the



least fixed point of the operator $\mathcal{B}$ applied on $\{V \in \mathcal{V} \mid V(s_\perp) = 0\}$ [Hartmanns and Kaminski, 2020].

Finally, notice that when we consider the values of a policy π , i.e., the values in the MC $\mathcal{M}_\pi$, the equations coincide when $\text{opt} = \max$ and $\text{opt} = \min$ and the solution is guaranteed to be unique.

^aPrecisely, the computation of the (undiscounted) expected return in $\mathcal{M}'$ is an instance of the *stochastic longest/shortest path problem* to the absorbing state $s_\perp$ [Bertsekas and Tsitsiklis, 1991].

Linear programming. Although the solution is not necessarily unique in the case where $\text{opt} = \max$, it is guaranteed to be the *least solution* of the equation. This property ensures the uniqueness of the solution of the following linear program [Forejt et al., 2011]:

$$\begin{aligned}
 &\textbf{Minimize} && \text{sign} \cdot \sum_{s \in S_\tau} v_s \\
 &\textbf{Subject to} && v_s \triangleright \sum_{s' \in S} \mathbb{P}(s' \mid s, a) \cdot v_{s'} && \text{for all } s \in S_\tau, a \in \mathcal{A} \\
 &&& v_s = 1 && \text{for all } s \in S_{=1} \\
 &&& v_s = 0 && \text{for all } s \in S_{=0},
 \end{aligned} \tag{2.29}$$

where $\text{sign} \in \{1, -1\}$ and $\triangleright \in \{\leq, \geq\}$ are respectively the sign and inequality relation. Their value differ according to the optimization direction: $\text{sign} = 1$ whenever $\text{opt} = \max$ and $\text{sign} = -1$ if $\text{opt} = \min$, whereas $\triangleright$ is equal to $\geq$ if $\text{opt} = \max$ and to $\leq$ if $\text{opt} = \min$. The unique solution v_s coincides with $\sup_\pi \mathbb{P}_\pi(\{\tau \in \text{Traj}_\mathcal{M} \mid \tau \models \Diamond a\} \mid s_I = s)$ whenever $\text{opt} = \max$ and $\inf_\pi \mathbb{P}_\pi(\{\tau \in \text{Traj}_\mathcal{M} \mid \tau \models \Diamond a\} \mid s_I = s)$ when $\text{opt} = \min$, for all $s \in S$. Since Markov chains have only one action, the LP can be further simplified by removing the action space from the constraints when checking the values of a policy π in the induced MC $\mathcal{M}_\pi$, which again reduces to a simple system of linear equations (see the related discussion in Section 1.6). Because linear programs can be solved in polynomial time, the subsequent theorem follows.

Theorem 10: Synthesis of policies for reachability properties

Given an atomic proposition $a \in \mathbf{AP}$ and an optimization direction $\text{opt} \in \{\max, \min\}$, an optimal deterministic stationary policy π^* defined for $\mathcal{M}$ which satisfies

$$\pi^* = \begin{cases} \sup_\pi \mathbb{P}_\pi(\tau \mid \tau \models \Diamond a) & \text{if } \text{opt} = \max \\ \inf_\pi \mathbb{P}_\pi(\tau \mid \tau \models \Diamond a) & \text{if } \text{opt} = \min \end{cases}$$

can be constructed in polynomial time in the size of $\mathcal{M}$.

Details on this construction can be found in [Baier and Katoen, 2008]. Essentially, this boils down to extracting the actions that maximize (resp. minimize) the value of Equation 2.28 for each state $s \in \mathcal{S}_\gamma$. Nevertheless, due to the absence of discounting, this extraction should be carefully realized to avoid the selection of actions possibly inducing dead cycles. Again, a suitable solution lies in conducting a graph analysis of the underlying MDP structure.

Computing the solution in practice. The Bellman’s equation can be computed via an LP, but as discussed in Section 1.6, LP solvers do not scale well to large problems and are rather slow in practice. Alternatively, one can opt for VI (Algorithm 1) or PI (Algorithm 2), even though these methods may exhibit exponential worst-case behavior, akin to the scenario involving (discounted) expected return. Nevertheless, in practice, they tend to converge quickly to the solution (refer to the final discussion of Section 1.6).

It is, however, worth noting that in contrast to discounted objectives, the Bellman operator is not a contraction mapping when the optimization direction is max. Although this does not impact the convergence of PI — as PI evaluates a distinct policy at each step, the Bellman operator is, in that case, a contraction mapping (cf. Remark 9) — the absence of discounting introduces considerably more intricate termination criteria for VI [Haddad and Monmege, 2014; Quatmann and Katoen, 2018; Hartmanns and Kaminski, 2020].

3.3 Stochastic Model Checking of LTL Properties

In the following, we describe a general procedure to model-check any arbitrary LTL formula φ in an MDP $\mathcal{M}$. The procedure is based on the translation of the LTL formula into a *deterministic Rabin automaton*, which allows generating ω -regular properties.

Intuitively, automata are finite-state machines defined over a finite input alphabet. Each automaton is equipped with an acceptance condition and an initial state. Input words are accepted by the automaton if and only if the specified acceptance condition is satisfied when the word is processed through the state machine from its initial state. The set of words accepted by the automaton is called the *language* of the automaton. The family of ω -regular properties is the set of all properties over infinite words, i.e., words of infinite length.

In our context, the input alphabet of automata consists of all the labels of the MDP. Then, such automata process traces of the MDP (in particular, the labels of states encountered along trajectories). When considering a specific formula φ , we are interested in automata whose language corresponds to the set of all traces that agree with φ . In particular, we consider *Rabin automata*, whose *acceptance condition* consists of a set of n pairs of states of the form

$$\{ \langle L_i, K_i \rangle \mid 1 \leq i \leq n \},$$

where states from L_i should be visited *finitely often* whereas those of K_i should be visited *infinitely often*. Formally, a deterministic Rabin automaton is defined as follows.

Definition 20: Deterministic Rabin Automaton

A *deterministic Rabin automaton* (DRA) is a tuple $\mathcal{A} = \langle \mathcal{Q}, \Sigma, \delta, q_i, \text{Acc} \rangle$ where

- $\mathcal{Q}$ is a finite set of states;
- Σ is a finite alphabet of symbols;
- $\delta : \mathcal{Q} \times \Sigma \rightarrow \mathcal{Q}$ is the transition function;
- $q_i \in \mathcal{Q}$ is the initial state; and
- $\text{Acc} = \{ \langle L_i, K_i \rangle \mid 1 \leq i \leq n \}$ is the acceptance condition, where $L_i, K_i \subseteq \mathcal{Q}$ for all $1 \leq i \leq n$.

For every word $w = \sigma_0 \sigma_1 \sigma_2 \dots \in \Sigma^\omega$, there is a *unique* sequence $q_0 \xrightarrow{\sigma_0} q_1 \xrightarrow{\sigma_1} q_2 \xrightarrow{\sigma_2} \dots$ in $\mathcal{A}$ so that $q_0 = q_i$, $q_t \in \mathcal{Q}$, and $\delta(q_t, \sigma_t) = q_{t+1}$ for each $t \geq 0$. The word w is accepted by $\mathcal{A}$ if and only if there exists a pair $\langle L_i, K_i \rangle \in \text{Acc}$ so that $q_t \in L_i$ for *finitely* many $t \geq 0$ and $q_{t'} \in K_i$ for *infinitely* many $t' \geq 0$. The set of infinite words accepted by $\mathcal{A}$ is called the *language* of $\mathcal{A}$, denoted $\mathcal{L}(\mathcal{A})$.

Example 12: DRA encoding an LTL formula

Let $\varphi = \Box(\heartsuit \wedge \blacklozenge \heartsuit)$ and $\mathcal{A}^\varphi$ be the DRA depicted in Figure 2.9. The DRA $\mathcal{A}^\varphi$ encodes the LTL formula φ : the acceptance condition requires the state q_0 to be visited infinitely often, which is returned every time the atomic propositions $\heartsuit$ and $\blacklozenge$ are eventually read.

Theorem 11: DRA and ω -regular properties

The class of languages accepted by DRAs is equivalent to the class of ω -regular languages [Grädel et al., 2002].

Furthermore, take $\Sigma = 2^{\text{AP}}$. Then, for any LTL formula φ over AP , there exists a DRA $\mathcal{A}^\varphi$ over Σ which encodes the property φ . The resulting DRA $\mathcal{A}^\varphi$ has a number of states and transitions which is, in the worst case, of order doubly exponential in the size of φ (2^{2^n} , where $n = |\varphi|$) [Vardi and Wolper, 1994; Daniele et al., 1999; Baier and Katoen, 2008].

Notice that every word accepted by a DRA with input alphabet $\Sigma = 2^{\text{AP}}$ is a sequence of labels. When considering the LTL formula φ translated into a DRA $\mathcal{A}^\varphi$, a word $w \in \Sigma^\omega$ is accepted by $\mathcal{A}^\varphi$ if the sequence of labels defined through w leads to the satisfaction of

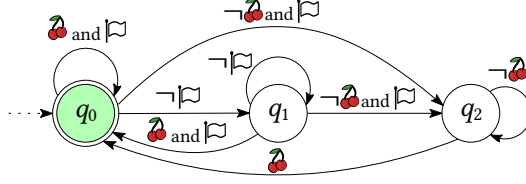


Figure 2.9: Deterministic Rabin Automaton (DRA) for the formula $\varphi = \Box(\Diamond a \wedge \Diamond b)$. The input alphabet is $\Sigma = 2^{\text{AP}} = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$ and the initial state is $q_i = q_0$. The edges represent the transitions of the automaton: an edge with an atomic proposition $a \in \text{AP}$ indicates the transition occurs upon reading a label embedding a ; with $\neg a$, the transition occurs if the label *does not embed* a ; and $a_1 \text{ and } a_2$ means that the transition occurs when reading a label embedding both $a_1, a_2 \in \text{AP}$. The acceptance condition is $\text{Acc} = \{ \langle L, K \rangle \}$ where $L = \emptyset$ and $K = \{q_0\}$ (in green).

the formula φ . Consequently, for all trajectories $\tau = \langle s_0 : \infty, a_0 : \infty \rangle \in \text{Traj}_{\mathcal{M}}$ and associated word $w = l_0 l_1 l_2 \dots \in 2^{\text{AP}}$ so that $l_i = \ell(s_i)$ for all $i \geq 0$,

$$\tau \models \varphi \quad \text{if and only if} \quad w \in \mathcal{L}(\mathcal{A}^\varphi).$$

Specifically, this suggests an algorithm to model-check any LTL formula φ in an MDP $\mathcal{M}$:

- (i) construct the DRA $\mathcal{A}^\varphi$ which encodes φ ;
- (ii) encode the DRA $\mathcal{A}^\varphi$ and its dynamics into the MDP $\mathcal{M}$, which results into some refined MDP $\mathcal{M}^{\mathcal{A}^\varphi}$; and
- (iii) compute the probability of satisfying the acceptance condition of $\mathcal{A}^\varphi$ in $\mathcal{M}^{\mathcal{A}^\varphi}$.

Concerning step (i), the construction of the DRA $\mathcal{A}^\varphi$ is highly non-trivial and is out of the scope of this thesis. We refer the interested reader to [Vardi and Wolper, 1994; Daniele et al., 1999; Baier and Katoen, 2008] for more details. Furthermore, several software tools are available to automatically construct the DRA $\mathcal{A}^\varphi$ from any LTL formula φ (see the survey of Blahoudek et al., 2013 for a non-exhaustive comparison of tools), one of the most recent and widely adopted being RABINIZER 4 [Kretínský et al., 2018].

For step (ii), the construction of the MDP $\mathcal{M}^{\mathcal{A}}$ for any arbitrary DRA $\mathcal{A}$, whose alphabet is over the atomic proposition space of $\mathcal{M}$, is defined as follows.

Definition 21: Product of an MDP with a DRA

Let $\mathcal{M} = \langle S, \mathcal{A}, \mathbf{P}, \mathcal{R}, \mathbf{AP}, \ell, s_I, \gamma \rangle$ be an MDP and $\mathcal{A} = \langle \mathcal{Q}, \Sigma, \delta, q_I, \text{Acc} \rangle$ be a DRA with alphabet $\Sigma = 2^{\mathbf{AP}}$ and acceptance condition $\text{Acc} = \{\langle L_i, K_i \rangle \mid 1 \leq i \leq n\}$. The product of $\mathcal{M}$ with $\mathcal{A}$ is the MDP

$$\mathcal{M}^{\mathcal{A}} = \langle S \times \mathcal{Q}, \mathcal{A}, \mathbf{P}^{\mathcal{A}}, \mathcal{R}^{\mathcal{A}}, \mathbf{AP}', \ell^{\mathcal{A}}, \langle s_I, q_I \rangle, \gamma \rangle$$

where for any $s, s' \in S, q, q' \in \mathcal{Q}$, and $a \in \mathcal{A}$,

- $\mathbf{P}^{\mathcal{A}}$ embeds the dynamics of both $\mathbf{P}$ and δ :

$$\mathbf{P}^{\mathcal{A}}(s', q' \mid s, q, a) = \begin{cases} \mathbf{P}(s' \mid s, a) & \text{if } q' = \delta(q, \ell(s')) \\ 0 & \text{otherwise;} \end{cases}$$

- $\mathcal{R}^{\mathcal{A}}$ is the projection of $\mathcal{R}$ on the state space of the MDP:

$$\mathcal{R}^{\mathcal{A}}(s, q, a) = \mathcal{R}(s, a);$$

- the space of atomic propositions embeds the acceptance condition of the DRA:

$$\bigcup_{1 \leq i \leq n} \{L_i\} \cup \{K_i\} \subseteq \mathbf{AP}'; \text{ and}$$

- the labeling function $\ell^{\mathcal{A}}$ allows to record the acceptance condition of the DRA into the product MDP:

$$\ell^{\mathcal{A}}(s, q) = \ell(s) \cup \{L_i : q \in L_i \text{ and } i \in \{1, \dots, n\}\} \cup \{K_i : q \in K_i \text{ and } i \in \{1, \dots, n\}\}.$$

Example 13: Product of an MC with a DRA

Consider the MDP $\mathcal{M}$ of Figure 2.8 and the LTL formula $\varphi = \Box(\Diamond \text{red} \wedge \Diamond \text{blue})$. For the sake of illustration, we fix the deterministic stationary policy π whose chosen actions are highlighted in blue in Figure 2.10a. Intuitively, the decisions stemming from π consist in leading the agent  towards  or , and then to continuously follow the path behind  introduced in Example 11. The DRA $\mathcal{A}^\varphi$ encoding φ is depicted in Figure 2.9. The product of $\mathcal{M}_\pi$ with $\mathcal{A}^\varphi$ is the MC $\mathcal{M}_\pi^{\mathcal{A}^\varphi}$ depicted in Figure 2.10.

The final step (iii) consists in computing the probability of fulfilling the acceptance condition of $\mathcal{A}^\varphi$ in $\mathcal{M}^{\mathcal{A}^\varphi}$, i.e., the probability of visiting states labeled with L_i finitely

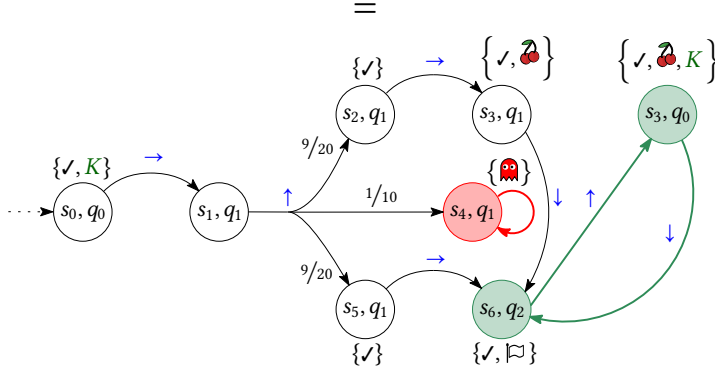
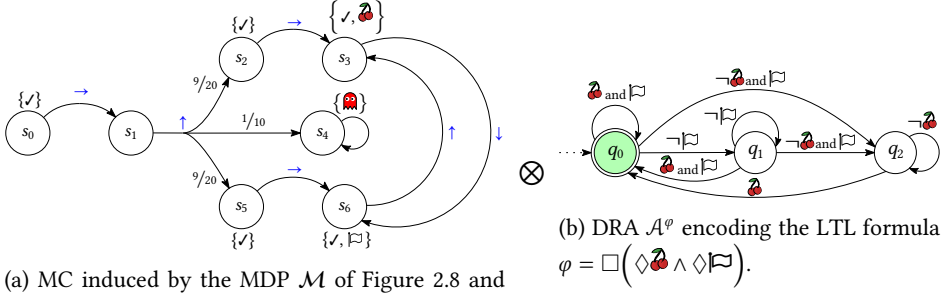


Figure 2.10: Product of the DRA $\mathcal{A}^\varphi$ with the MC $\mathcal{M}_\pi$ (unreachable states are omitted). The BSCCs (Definition 9) of the resulting product are highlighted in red and green. The probability of satisfying the LTL formula φ under the policy π is equal to the probability of visiting infinitely often states labeled with K . In particular, the BSCC highlighted in green, $B = \{\langle s_3, q_0 \rangle, \langle s_6, q_2 \rangle\}$, is the only one containing a state labeled with K (precisely, the state $\langle s_3, q_0 \rangle$), where all states are repeated infinitely often with probability one. Therefore, the probability of satisfying φ is equal to the probability of reaching B .

often and those labeled with K_i infinitely often, for all pairs where $1 \leq i \leq n$. This boils down to computing the probability of satisfying the following LTL formula in $\mathcal{M}^{A^g}$:

$$\bigvee_{1 \leq i \leq n} (\Diamond \Box \neg L_i \wedge \Box \Diamond K_i).$$

Note that the event $\{\tau \in \text{Traj}_{\mathcal{M}^{A^g}} \mid \tau \models \bigvee_{1 \leq i \leq n} (\Diamond \Box \neg L_i \wedge \Box \Diamond K_i)\}$ is measurable, since visiting a set of states infinitely often can be formalized as countable intersections and unions of cylinder sets (Section 1.4, Equation 2.5), as is the event of visiting a set of states finitely often since the latter is the complement of visiting them infinitely often.

Reachability is all you need. Let $\text{inf}(\tau)$ be the set of states visited infinitely often along the trajectory $\tau \in \text{Traj}_{\mathcal{M}^{A^g}}$, as defined in Section 1.4 (Equation 2.4). For any $1 \leq i \leq n$, detecting whether

- (a) L_i is visited *finitely often* along τ consists in checking whether *no state* from $\text{inf}(\tau)$ is labeled with L_i , i.e., if $\text{inf}(\tau) \cap \{ \langle s, q \rangle \in S \times Q \mid L_i \in \ell^A(s, q) \} = \emptyset$; and
- (b) K_i is visited *infinitely often* along τ consists in checking whether *there is some state* in $\text{inf}(\tau)$ that is labeled with K_i , i.e., $\text{inf}(\tau) \cap \{ \langle s, q \rangle \in S \times Q \mid K_i \in \ell^A(s, q) \} \neq \emptyset$.

Recall now that $\text{inf}(\tau)$ almost surely consists of a BSCC (Equation 2.6) for any infinite trajectory τ . Then, checking (a) and (b) under any policy π boils down to *computing the reachability probability to the BSCCs of $\mathcal{M}_\pi^{A^g}$ where some state is labeled with K_i while no state is labeled with L_i* . The synthesis procedure is similar: one needs to *detect* the regions of the system containing the possible BSCCs of $\mathcal{M}^{A^g}$, *classify* those regions according to their state labels, and then (i) construct a policy which maximizes (resp. minimize) the probability to reach the regions fulfilling the acceptance condition, and (ii) construct a policy in each of these regions which continuously allows visiting states labeled with K_i , while avoiding those labeled with L_i .

Limiting regions of finite MDPs. Those particular regions of the system that contain all the possible BSCCs are called *maximal end components*.

Definition 22: End Component

An *end component* (EC) of $\mathcal{M}$ is a tuple $\langle B, A \rangle$ where $B \subseteq S$ and $A : B \rightarrow 2^A$ is a function which assigns to each state its *set of enabled actions* in B that satisfies the following condition. Under any stationary policy π whose support agrees with A , i.e., $\text{supp}(\pi(\cdot \mid s)) = A(s)$ for all $s \in B$, the set B is a BSCC of $\mathcal{M}_\pi$.

Note that a single EC with state space B may embed several BSCCs: a policy whose support does not necessarily agree with all the enabled actions of the EC but which is still included in the set of enabled actions of each state may induce a BSCC which is included in B . Then, we say that an EC $\langle B_1, A_1 \rangle$ is *included* in another EC $\langle B_2, A_2 \rangle$ if $B_1 \subseteq B_2$ and $A_1(s) \subseteq A_2(s)$ for all $s \in B_1$. Furthermore, the union of two ECs with $B_1 \cap B_2 \neq \emptyset$ is also an EC. This leads to the notion of *maximal* end component.

Definition 23: Maximal End Component

A *maximal end component* (MEC) is an EC $\langle B, A \rangle$ which cannot be extended: there is no EC $\langle B', A' \rangle$ with $B \subset B'$ or $A(s) \subset A'(s)$ for some $s \in B$.

While the number of ECs may be in the worst case exponential in the size of $\mathcal{M}$ (i.e., in the number of states and actions in $\mathcal{M}$), any EC is included in exactly one MEC. Furthermore, all MECs are pairwise disjoint, so the number of MECs is bounded by the number of states in $\mathcal{M}$ [Baier and Katoen, 2008]. Note that when a policy π is fixed in $\mathcal{M}$, the MECs of $\mathcal{M}_\pi$ are exactly the BSCCs of $\mathcal{M}_\pi$.

Detecting the BSCCs of $\mathcal{M}$ under a specific policy π or the MECs of $\mathcal{M}$ can once again purely be done via graph analysis algorithms. Precisely, this detection can be realized via a strongly connected component decomposition of the underlying graph of the state-transition structure of $\mathcal{M}^{\mathcal{A}^\varphi}$ [Baier and Katoen, 2008]. Figure 2.11 illustrates an example of MEC decomposition in an MDP.

Classification of accepting regions. Recall that *all states* of a BSCC induced by a policy are repeated infinitely often with probability one. For a given DRA $\mathcal{A}^\varphi$, we aim at determining whether the BSCCs/MECs, with a set of states B , are accepted or not based on the pair $\langle L_i, K_i \rangle$. This is done by verifying if the given policy π allows returning to each state $\langle s, q \rangle$ marked with K_i with probability one. This return should only occur through states of B that do not carry the label L_i . If this is the case, then the region is classified as fulfilling the acceptance condition of $\mathcal{A}^\varphi$.

Precisely, when the policy π is fixed, the classification merely consists of checking whether there is no state labeled with L_i , while at least one state is labeled with K_i in the BSCC B . For MECs, the classification is slightly more complex. Indeed, one needs to check whether at least one EC included in a given MEC $\langle B, A \rangle$ fulfills the latter conditions. This can be done by adding the atomic proposition $\mathbf{a}_{s,q}$ to AP for all $\langle s, q \rangle \in B$ satisfying $K_i \in \ell^{\mathcal{A}^\varphi}(s, q)$. Then, it remains to check if there exists a policy π which satisfies with probability one the property

$$\mathbf{a}_{s,q} \wedge \circ (\neg L_i \cup \mathbf{a}_{s,q}).$$

from some state $\langle s, q \rangle$ labeled with K_i . Checking the probability of a reach-avoid property of the form $\neg L_i \cup \mathbf{a}_{s,q}$ can be merely performed by making each state labeled with L_i

absorbing, and then checking the reachability probabilities to $\mathbf{a}_{s,q}$ in the resulting model. The next operator $\circ$ simply consists in checking the probability of transitioning to the successor states from which the reach-avoid property is satisfied. The full LTL model checking algorithm is detailed in Algorithm 5. The synthesis procedure — constructing the policy which maximizes (resp. minimizes) the probability of satisfying φ — consists in constructing the Mealy machine which encodes π (cf. Property 1) from the value functions computed in Algorithm 5.

LTL synthesis. Let π be a policy encoded by the Mealy machine $\langle Q, \pi_a, \pi_u, q_I \rangle$, then π maximizes (resp. minimizes) the probability of satisfying φ if

- (i) the state space $\mathcal{Q}^\varphi$ of the DRA $\mathcal{A}^\varphi$ is encoded into the memory of π as well as the transition function δ of the DRA into the policy update function: $Q = \mathcal{Q}^\varphi$ and for any action $a \in \mathcal{A}$ as well as pair of states $s \in S, q \in Q$,

$$\pi_u(q' \mid s, q, a) = \begin{cases} 1 & \text{if } q' = \delta(q, \ell(s)) \\ 0 & \text{otherwise;} \end{cases}$$

- (ii) let $\pi^{\diamond\checkmark}$ be a policy that maximizes (resp. minimizes) the probability of reaching states labeled with $\checkmark$ in $\mathcal{M}^{\mathcal{A}^\varphi}$, which can be constructed according to the value vector $V_{\diamond\checkmark}$ computed in Algorithm 5 (cf. Theorem 10). Then, the decisions of $\pi^{\diamond\checkmark}$ are encoded into the policy: when a state $s \in S$ is visited while the policy is in memory state $q \in Q$, the DRA has for current state $q' = \delta(q, \ell(s))$.¹³ So, if $\langle s, q' \rangle$ is *not labeled with $\checkmark$* ,

$$\pi_a(\cdot \mid s, q) = \pi^{\diamond\checkmark}(\cdot \mid s, q');$$

- (iii) on the other hand, if $\langle s, q' \rangle$ is labeled with $\checkmark$, the decisions of π depend on the MEC $\langle B, A \rangle$ which contains $\langle s, q' \rangle$, as well as the Rabin pair $1 \leq i \leq n$ which is accepted in $\langle B, A \rangle$. Precisely, let V_B^i be the value vector computed in Algorithm 5 for the MEC $\langle B, A \rangle$ satisfying the Rabin pair $1 \leq i \leq n$, $B_{=1}^i = \{ \langle s^*, q^* \rangle \in B \mid V_B^i(s^*, q^*) = 1 \}$ be the related almost sure reachability set, and π_B^i be the policy which is constructed according to V_B^i . Then, for all $a \in \mathcal{A}$, the decisions of π_B^i are encoded into π via

$$\pi_a(a \mid s, q) = \begin{cases} \pi_B^i(a \mid s, q') & \text{if } q' \notin K_i \\ 1 & \text{else, if } a = \arg \text{opt}_{a^* \in \mathcal{A}(s, q')} \mathbf{P}^{\mathcal{A}^\varphi}(B_{=1}^i \mid s, q', a^*) \\ 0 & \text{otherwise.} \end{cases}$$

¹³Notice the slight asymmetry which results from encoding δ into the memory update function π_u : while the state of the DRA is updated according to the label of the next state visited, the Mealy machine can only be updated according to the label of the current state. This results from the fact that the Mealy machine is updated before that the transition to the next MDP state occurs, while the DRA is only updated after that this transition is performed. Then, π_a needs to take into account this delay, which is resolved by syncing its memory q with the one of the DRA at decision time, via $q' = \delta(q, \ell(s))$ where s is the current state of $\mathcal{M}$.

Algorithm 5: LTL Model Checking

Input:

- LTL formula φ ;
- Special symbol $\checkmark \in \mathbf{AP}$ so that there is no state $s \in S$ where $\checkmark \in \ell(s)$;
- Solver $\mathbf{reach}_{\mathcal{M}}^{\text{opt}} : \mathbf{AP} \rightarrow [0, 1]$ for the reachability problem (Theorem 9) in the optimization direction opt for the MDP $\mathcal{M}$ (e.g., linear programming, VI, or PI);
- (optional) Finite-memory policy π ; and
- (optional) Optimization direction $\text{opt} \in \{\max, \min\}$ (default: $\max$).

if π *is provided* **then**

$\mathcal{M} \leftarrow \mathcal{M}_{\pi}$

Translate φ into a DRA $\mathcal{A}^{\varphi}$ with acceptance condition $\text{Acc} = \{\langle L_i, K_i \rangle \mid 1 \leq i \leq n\}$

Do the product between $\mathcal{M}$ and $\mathcal{A}^{\varphi}$, resulting in the MDP $\mathcal{M}^{\mathcal{A}^{\varphi}}$

Detect the MECs of $\mathcal{M}$: $\mathbb{M} \leftarrow \{\langle B, A \rangle \mid \langle B, A \rangle \text{ is a MEC of } \mathcal{M}\}$

foreach $\langle B, A \rangle \in \mathbb{M}$ **do**

forall $i \leftarrow 1$ **to** n **do**

if π *is provided* or $A(s, q)$ *is a singleton for all* $\langle s, q \rangle \in B$ **then**

if $\{\langle s, q \rangle \in B \mid L_i \in \ell^{\mathcal{A}^{\varphi}}(s, q)\} = \emptyset$ *and*

$\{\langle s, q \rangle \in B \mid K_i \in \ell^{\mathcal{A}^{\varphi}}(s, q)\} \neq \emptyset$ **then**

Label B with $\checkmark$: $\ell^{\mathcal{A}^{\varphi}}(s, q) \leftarrow \ell^{\mathcal{A}^{\varphi}}(s, q) \cup \{\checkmark\}$ for all $\langle s, q \rangle \in B$

else

▷ *Make absorbing all states labeled with* L_i *in the MEC* $\langle B, A \rangle$

forall $\langle s, q \rangle \in B$ *and* $a \in A$ **do**

$\mathbf{P}_i(s', q' \mid s, q, a) \leftarrow \begin{cases} \mathbb{1}\{\langle s, q \rangle = \langle s', q' \rangle\} & \text{if } q \in L_i \text{ or } a \notin A(s, q) \\ \mathbf{P}^{\mathcal{A}^{\varphi}}(s', q' \mid s, q, a) & \text{otherwise} \end{cases}$

Let $\mathcal{M}_i$ be the MDP obtained by replacing $\mathbf{P}^{\mathcal{A}^{\varphi}}$ by $\mathbf{P}_i$

▷ *Check the probabilities of returning to each state labeled with* K_i *in* $\mathcal{M}_i$

foreach $\langle s, q \rangle \in B$ *labeled with* K_i **do**

$\mathbf{AP} \leftarrow \mathbf{AP} \cup \{a_{s,q}\}$; $\ell^{\mathcal{A}^{\varphi}}(s, q) \leftarrow \ell^{\mathcal{A}^{\varphi}}(s, q) \cup \{a_{s,q}\}$

Compute the values of reaching $\langle s, q \rangle$: $V_B^i \leftarrow \mathbf{reach}_{\mathcal{M}_i}^{\text{opt}}(a_{s,q})$

Almost sure reachability set: $B_{=1}^i \leftarrow \{\langle s', q' \rangle \in B \mid V_B^i(s', q') = 1\}$

▷ *Label* $B_{=1}^i$ *with* $\checkmark$ *if the probability of returning to* $\langle s, q \rangle$ *is one*

if $\text{opt}_{a \in A(s,q)} \mathbf{P}_i(B_{=1}^i \mid s, q, a) = 1$ **then**

$\ell^{\mathcal{A}^{\varphi}}(s^*, q^*) \leftarrow \ell^{\mathcal{A}^{\varphi}}(s^*, q^*) \cup \{\checkmark\}$ for all $\langle s^*, q^* \rangle \in B_{=1}^i$

$V_{\diamond\checkmark} \leftarrow \mathbf{reach}_{\mathcal{M}^{\mathcal{A}^{\varphi}}}^{\text{opt}}(\checkmark)$

return *Probability of satisfying* φ *from the initial state:* $V_{\diamond\checkmark}(s_I, q_I)$

Complexity. Notice that all operations in Algorithms 5 can be performed in polynomial time in the size of $\mathcal{M}^{\mathcal{A}^\varphi}$ (detecting the MECs, iterating over the states of each MEC, computing reachability probabilities, etc). However, $\mathcal{A}^\varphi$ is in the worst case doubly exponential in the size of the LTL formula φ (cf. Theorem 11). Then, the complexity of the LTL model checking algorithm presented in this section as well as the memory required by the synthesized policies are in the worst case of order doubly exponential in the size of φ . Considering more advanced kinds of automata, LTL model checking in Markov chains is actually shown to be in **PSPACE**, using *unambiguous automata* whose blow-up is single- (instead of double-) exponential [Baier et al., 2023]. In MDPs, the double exponential blow-up cannot be avoided in the worst case, but *limit-deterministic automata* can lead to a significantly smaller construction. Moreover, for some expressive fragments of LTL, the complexity can be reduced to **EXPTIME**, as opposed to **2EXPTIME** [Sickert et al., 2016; Kini and Viswanathan, 2017a,b].

Example 14: Stochastic LTL Model Checking and Synthesis

We illustrate the application of Algorithm 5 on the MDP $\mathcal{M}$ of Figure 2.8 for the LTL formula $\varphi = \Box(\Diamond \textcolor{red}{a} \wedge \Diamond \textcolor{blue}{b})$. A DRA encoding φ is depicted in Figure 2.9, where the acceptance condition merely consists of visiting infinitely often the set $K = \{q_0\}$. We first present an example where a policy is provided, and then, for completeness, we present the case for which no policy is provided.

Model checking an MC. Consider the deterministic stationary policy π whose decisions are highlighted in **blue** in Figure 2.10a. As illustrated in Figure 2.10, the product of $\mathcal{M}_\pi$ with $\mathcal{A}^\varphi$ is the MC $\mathcal{M}_\pi^{\mathcal{A}^\varphi}$. The BSCCs of $\mathcal{M}_\pi^{\mathcal{A}^\varphi}$, $B_1 = \{\langle s_4, q_1 \rangle\}$ and $B_2 = \{\langle s_6, q_2 \rangle, \langle s_3, q_0 \rangle\}$, are the sole regions of the state space of $\mathcal{M}_\pi^{\mathcal{A}^\varphi}$ where states are visited infinitely often with nonzero probability (and further, almost surely). Then, the probability of satisfying φ under π is equal to the probability of reaching any state belonging to the BSCC B_2 , since the latter contains $\langle s_3, q_0 \rangle$ which is labeled with K . In Figure 2.10, reaching the **green** region of the MC is clearly equal to $9/10$. Then,

$$\mathbb{P}_\pi^{\mathcal{M}}(\{\tau \mid \tau \models \varphi\}) = \mathbb{P}_\pi^{\mathcal{M}_\pi^{\mathcal{A}^\varphi}}(\Diamond B_2) = \frac{9}{10},$$

where $\Diamond B_2$ is the event of reaching the BSCC B_2 , i.e.,

$$\Diamond B_2 = \{s_0:\infty, q_0:\infty, a_0:\infty \in \text{Traj}_{\mathcal{M}_\pi^{\mathcal{A}^\varphi}} \mid \text{there is a } t \geq 0 \text{ so that } \langle s_t, q_t \rangle \in B_2\}.$$

Model checking an MDP. Now, consider that no policy is provided. The product of $\mathcal{M}$ with $\mathcal{A}^\varphi$ is the MDP $\mathcal{M}^{\mathcal{A}^\varphi}$ depicted in Figure 2.11. The next step consists of detecting the MECs of $\mathcal{M}^{\mathcal{A}^\varphi}$. The MECs are highlighted in different colors in

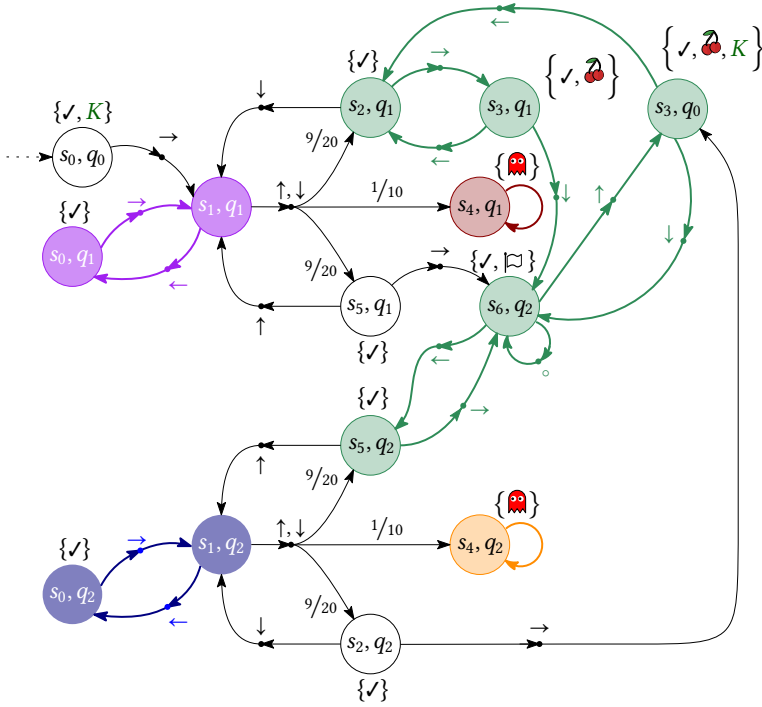


Figure 2.11: Product of the DRA $\mathcal{A}^\varphi$ of Example 12 with the MDP $\mathcal{M}$ of Figure 2.8 (unreachable states are omitted), resulting in the MDP $\mathcal{M}^{\mathcal{A}^\varphi}$. The MECs of $\mathcal{M}^{\mathcal{A}^\varphi}$ are highlighted in colors (respectively, in purple, blue, green, orange, and brown).

Figure 2.11. In particular, the MEC colored in **green** is the sole one that includes a state labeled with K . Then, since there is no state to visit finitely often in $\mathcal{A}^\varphi$, the maximum probability of satisfying φ is equal to the probability of reaching the MEC colored in **green**, which is equal to $9/10$, similarly to the previous example. Indeed, we know by definition of ECs (Definition 22) that some policy exists which allows to visit each state of this MEC infinitely often with probability one, then so is K .

On the other hand, the minimum probability of satisfying φ is induced by policies that allow to end up in BSCCs embedding no state labeled with K , which is achieved by any policy that reaches and allows to stay in the MECs colored in **purple**, **blue**, **orange**, and **brown**, or those that end up in the MEC colored in **green** but strictly avoid the state labeled with K . Precisely, the latter policies correspond to those that never choose the action $\uparrow$ in the state s_6 . All those policies yield a zero probability of satisfying φ since they were enforced to avoid the acceptance condition.

Synthesis. Synthesizing a policy that minimizes the probability of satisfying φ is straightforward: in s_1 , deterministically choosing the action $\leftarrow$ allows to stay in the MEC colored in **purple**, which does not contain any state labeled with K . There, the probability of satisfying φ is zero. More generally, any policy that allows to avoid the state labeled with K in the MEC colored in **green** yields a zero probability of satisfying φ .

For the construction of a policy that maximizes the probability of satisfying φ , one first needs to construct a policy that maximizes the probability of reaching the **green** area in $\mathcal{M}^{\mathcal{A}^\varphi}$. Then, once the **green** MEC is reached, the policy must choose some enabled actions of this MEC that allow continuously reaching the state labeled with K . In particular, any policy whose support agrees with this set of enabled actions — thus any policy which (probabilistically) chooses all actions highlighted in **green** — satisfies this requirement.

3.4 Bad News: LTL is not Learnable

Recall now the context of this thesis. We aim at verifying the agent's behaviors in the context of RL, where the model of the environment is not a priori known or explicitly given (cf. Section 2). To achieve this through model checking, the intuitive idea will be to learn a model of the environment from data collected by the agent, which will be formally verified to yield the desired guarantees afterward (cf. Chapter 1).

To provide any guarantee on the satisfaction of an LTL property φ via model checking, the values of φ should be at least *learnable* to consider verifying the policy in a learned model. Specifically, given a policy π for the agent and a desired error bound $\varepsilon > 0$ on the value, one should be able to estimate with a high probability the value of φ — or in other words, derive a *probably approximately correct* (PAC) estimation. The idea is to

accurately approximate the value of φ using an estimator constructed from a dataset of trajectories collected via a *learning algorithm*.

Definition 24: Learning algorithm

Given an MDP $\mathcal{M}$ with state space $\mathcal{S}$, action space $\mathcal{A}$, and a policy π , a *learning algorithm* $\mathcal{L}_\pi^\mathcal{M}$ is implemented as a distribution that allows collecting finite trajectories of $\mathcal{M}$ when π is executed. Precisely, $\mathcal{L}_\pi^\mathcal{M}$ is a probability measure $\mathbb{P}_{\pi'}^{\mathcal{M}'}$ induced by an MDP $\mathcal{M}'$ and a policy π' defined as follows:

- (i) $\mathcal{M}'$ is a copy of $\mathcal{M}$ whose state/action spaces are respectively augmented with $\mathcal{S} \cup \{s_{\text{reset}}\}$ and $\mathcal{A} \cup \{a_{\text{reset}}\}$: whenever a_{reset} is executed, the MDP is reset to s_{reset} , which transitions to the initial state s_I with probability one;
- (ii) π' mirrors the choice of π , but eventually produces a_{reset} ; and
- (iii) $\mathcal{M}'_{\pi'}$ is episodic (Definition 16).

Remark 10: Step-bounded learning algorithm

One may further require that the length of trajectories do not exceed some threshold $N \geq 0$. In that case, π' embeds in its memory a counter $n \geq 0$ which is incremented at each step of the interaction; π' mirror π whenever $n < N$ and produces a_{reset} when $n = N$. The counter is then reset to zero and the process restarts.

Intuitively, a learning algorithm $\mathcal{L}_\pi^\mathcal{M}$ allows simulating $\mathcal{M}$ under any policy π by almost surely periodically restarting the process. The episodes generated through such a learning algorithm consist of finite trajectories of $\mathcal{M}$ (possibly excluding the last state, s_{reset}). This process yields a dataset $\mathcal{D}$ comprising trajectory samples from $\mathcal{M}$. Importantly, as in RL, the learning algorithm does not have explicit access to the transition function $\mathbf{P}$ of $\mathcal{M}$: it samples trajectories by executing π and decides when to restart the simulation. If it is feasible to estimate the value of a random variable $\mathbb{O}$ from a finite dataset formed in this manner, then $\mathbb{O}$ is considered *PAC learnable*.

Definition 25: PAC Learnable Objective

Given any *objective function* $\mathbb{O} : \text{Traj}_{\mathcal{M}} \rightarrow \mathbb{R}$, and a policy π for $\mathcal{M}$, $\mathbb{O}$ is *probably approximately correctly* (PAC) *learnable* if and only if for any $\varepsilon > 0$ and $\delta \in (0, 1)$, there exists a bound $T \in \mathbb{N}$ and a learning algorithm $\mathcal{L}_{\pi}^{\mathcal{M}}$ so that *for all* collections of finite trajectories drawn from $\mathcal{L}_{\pi}^{\mathcal{M}}$ of the form

$$\mathcal{D} = \{ \tau_1, \dots, \tau_T : \tau_i \text{ is a prefix of some trajectory } \tau \sim \mathbb{P}_{\pi}^{\mathcal{M}} \text{ for all } 1 \leq i \leq T \},$$

there is an estimator $\mathcal{E}_{\mathcal{D}} \in \Delta(\text{Traj}_{\mathcal{M}})$ constructed from $\mathcal{D}$ so that

$$|\mathbb{E}_{\pi}^{\mathcal{M}}[\mathbb{O}] - \mathbb{E}_{\tau \sim \mathcal{E}_{\mathcal{D}}}[\mathbb{O}]| \leq \varepsilon$$

with probability at least $1 - \delta$.

In our context, given an LTL formula φ , the objective function $\mathbb{O}^{\varphi}$ merely consists of the indicator $\mathbb{1}\{\tau \models \varphi\}$. The dataset $\mathcal{D}$ is generated by simulating the policy π in $\mathcal{M}$ through the sampling algorithm $\mathcal{L}_{\pi}^{\mathcal{M}}$ during T episodes determined by the desired accuracy. Then, the estimator $\mathcal{E}_{\mathcal{D}}$ may, e.g., be defined as the empirical distribution P of the trajectories in the dataset $P(\tau) = 1/T \sum_{i=1}^T \mathbb{1}\{\tau_i \text{ is a prefix of } \tau\}$, or the measure $\mathbb{P}^{\mathcal{M}}$ of another MC $\hat{\mathcal{M}}$ constructed (or learned) from transitions stored in $\mathcal{D}$ [Bazille et al., 2020].

To summarize, the PAC learnability of φ consists in accurately estimating the probability of satisfying φ from *any* finite dataset $\mathcal{D}$ formed by executing π in $\mathcal{M}$ according to some learning algorithm $\mathcal{L}_{\pi}^{\mathcal{M}}$, where the size of $\mathcal{D}$ is determined by both ε and δ . Unfortunately, this is not possible for any arbitrary LTL formula φ , as stated in the following theorem.

Theorem 12: LTL is not PAC Learnable [Bazille et al., 2020; Yang et al., 2022]

Given any policy π , there is some LTL formula φ so that, whatever the accuracy requested via $\varepsilon > 0$ and $\delta \in (0, 1)$, there is no learning algorithm $\mathcal{L}_{\pi}^{\mathcal{M}}$ which allows generating an estimator $\mathcal{E}_{\mathcal{D}}$ with a finite number of sample trajectories so that

$$|\mathbb{P}_{\pi}^{\mathcal{M}}(\{\tau \mid \tau \models \varphi\}) - \mathcal{E}_{\mathcal{D}}(\{\tau \mid \tau \models \varphi\})| \leq \varepsilon$$

with at least probability $1 - \delta$.^a

^awe note that the result was first proven by Bazille et al. in the context of Markov chains. Similar arguments were then used by Yang et al. for learning optimal values in MDPs with general estimators.

Note that the fact that it is not possible to design an estimator for a particular formula implies that it is all the more impossible to design an estimator that allows to probably approximately check the satisfaction of every LTL formula.

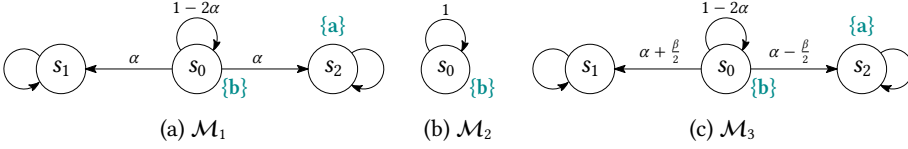


Figure 2.12: Markov chains $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3$ illustrating the intractability of learning the probability of satisfying the LTL formula $\Diamond a$ from data, where $0 < \beta < 2\alpha < 1$.

The intuition behind this intractability is that when the number of possible interactions is (almost surely) bounded, an environment with unknown dynamics may be indistinguishable from another one with different dynamics, where the satisfaction probability of LTL formulae might be arbitrarily different. To illustrate this, consider the Markov chains $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3$ depicted in Figure 2.12 (from the proof of Bazille et al., 2020). Note that

$$\begin{aligned} \mathbb{P}^{\mathcal{M}_1}(\{\tau \mid \tau \models \Diamond a\}) &= \alpha + (1 - 2\alpha) \cdot \mathbb{P}^{\mathcal{M}_1}(\{\tau \mid \tau \models \Diamond a\}) = \frac{1}{2}, \\ \mathbb{P}^{\mathcal{M}_2}(\{\tau \mid \tau \models \Diamond a\}) &= 0, \text{ and} \\ \mathbb{P}^{\mathcal{M}_3}(\{\tau \mid \tau \models \Diamond a\}) &= \alpha - \frac{\beta}{2} + (1 - 2\alpha) \cdot \mathbb{P}^{\mathcal{M}_3}(\{\tau \mid \tau \models \Diamond a\}) = \frac{2\alpha - \beta}{4\alpha} = \frac{1}{2} - \frac{\beta}{4\alpha}. \end{aligned}$$

Assume that $\mathcal{M}_1$ is the original MC, and recall that its transition function is unknown to the learning algorithm $\mathcal{L}^{\mathcal{M}_1}$. Recall that the value of α is unknown. Therefore, whatever the error terms ε, δ , the learning algorithm $\mathcal{L}^{\mathcal{M}_1}$ cannot be designed according to α . Then, since the number of transitions collected in $\mathcal{D}$ is finite, a sufficiently small α may lead to a high probability of exclusively observing the transition $s_0 \rightarrow s_0$ during the interaction. This deception could further mislead the learning algorithm $\mathcal{L}^{\mathcal{M}_1}$ into believing it is operating within $\mathcal{M}_2$. This happens when α^{-1} is large enough compared to the total number of transitions collected in $\mathcal{D}$. In that case, the error is arbitrarily large (0 vs. $1/2$).

In fact, even if one incorporates information about the underlying structure of $\mathcal{M}_1$ into $\mathcal{L}_\pi^{\mathcal{M}_1}$, accounting for the existence of transitions $s_0 \rightarrow s_1$ and $s_0 \rightarrow s_2$, the error might still be arbitrarily large. To understand why, consider $\varepsilon > 0$ as the desired error for the probability of reaching a . While $\mathcal{L}_\pi^{\mathcal{M}_1}$ may gather enough data to witness an arbitrarily small error $\beta/2$ on the transition $s_0 \rightarrow s_2$, it may then assume it operates within $\mathcal{M}_3$. However, it is possible that α is so small that $\frac{\beta}{2(1-\varepsilon)} > \alpha > \frac{\beta}{2}$. In this situation,

$$\mathbb{P}^{\mathcal{M}_3}(\{\tau \mid \tau \models \Diamond a\}) = \frac{1}{2} - \frac{\beta}{4\alpha} < \frac{1}{2} - \frac{2(1-\varepsilon)\beta}{4\beta} = \frac{1}{2} - \frac{1-\varepsilon}{2} = \frac{\varepsilon}{2},$$

leading to an arbitrarily large error ($\varepsilon/2$ vs. $1/2$).

The fact that we cannot ensure a PAC algorithm for reachability properties breaks the possibility of PAC-estimating the satisfaction probabilities of LTL formulae in general since the computation of their probability relies on reachability properties (cf. Section 3.3).

3.5 Valued λ -calculus

The intractability of learning the probability of satisfying LTL formulae from data suggests that we need to tackle the problem of verifying RL policies from a different perspective and consider other learnable objectives. In particular, one could consider working with LTL over finite trajectories (LTL_f, Camacho et al., 2019) instead of LTL, which yields simple algorithms, where automata over finite words suffice for checking properties [Wells et al., 2020]. However, this comes at the cost of losing the expressiveness of LTL. For example, specifications like $\Diamond\Box a$ and $\Box\Diamond a$ may both induce the same (undesirable) behavior since a merely needs to hold at the end of the finite trajectory to satisfy both specifications. This simplified interpretation might be misleading.

Discounting. The primary obstacle hindering the PAC learnability of LTL specifications lies in the inability to distinguish trajectories that are arbitrarily close. Consider two trajectories $\tau_1, \tau_2 \in \text{Traj}_{\mathcal{M}}$ that are identical up to a certain *large* time step $t > 0$ but may diverge from t onwards. The length of the shared prefix up to time t is irrelevant, as both trajectories either satisfy or violate the formula φ , resulting in a binary payoff incurred by the objective function $\mathbb{O}^\varphi(\cdot) \in \{0, 1\}$. This stands in contrast to discounting objectives, like the discounted return, where the discount factor decays to the extent that the discrepancy between trajectories beyond time t becomes negligible.

Moreover, this boolean interpretation of the satisfaction of LTL formulae does not enable to specify *how fast* or *how well* a specification is satisfied by the agent. First, consider the formula $\Diamond a$. In $\mathcal{M}_1$ (Figure 2.12), any trajectory of the form $\tau = (s_0 \rightarrow)^t s_2^\omega$ is assigned to a payoff of $\mathbb{O}^{\Diamond a}(\tau) = 1$, whatever the value of t . Yet we have seen that t might be arbitrarily large as long as α is sufficiently small, and there is no cost put on trajectories that reach **a** after a long time compared to those which does reach the target quickly. Second, consider the formula $\Box b$. In $\mathcal{M}_1$, $\mathbb{O}^{\Box b}(\tau)$ almost surely outputs zero, whatever the length t of the prefix of τ . However, we have seen that it might be the case that α is so small that even a learning algorithm cannot distinguish between $\mathcal{M}_1$ and $\mathcal{M}_2$, and no “reward” is put on the fact that the agent stays in **b** for a long time.

Valued λ -calculus. To address such issues, we consider *real valued* functions, which assign to each state a value between 0 and 1. Such functions capture the intuitive idea that events occurring early during the interaction are more important than those occurring

later. Specifically, we introduce the *valued λ -calculus* (VLC), a logical formalism that allows to give a *discounted* interpretation to classical specification formalisms.

VLC is an instance of the *quantitative μ -calculus* of de Alfaro et al. [2003], originally introduced in the context of concurrent probabilistic game structures, where discounts are not necessarily strictly lower than one. In quantitative μ -calculus, the semantics consist of computing the least and greatest fixed points of functions mapping states to real values (respectively leading to two fixed points operators: μ and ν). Here, we assume the discounts to be strictly inferior to one, leading to contraction mappings and the uniqueness of the fixed points. In that case, one need not distinguish between the least and greatest fixed points operators, and we simply denote the related operator by λ .

Beyond the discounted interpretation, contraction mappings imply the computability of the fixed points through the Banach fixed point theorem, enabling techniques like VI to compute the values of the formulae. Furthermore, there is a connection between general ω -regular properties (therefore including LTL) and VLC formulae, which allows expressing a discounted version of the former in the latter [de Alfaro et al., 2003]. We further establish the PAC learnability of the values of such linear formulae.

Definition 26: Syntax of valued λ -calculus

The syntax of VLC over the set of atomic propositions $\mathbf{AP}$ and set of variables $\mathbb{X}$ is defined by the following grammar:

$$\varphi := \mathbf{a} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \gamma \cdot \circ^{\text{opt}}\varphi^* \mid (1 - \gamma) + \gamma \cdot \circ^{\text{opt}}\varphi^* \mid \lambda x. \varphi$$

$$\varphi^* := x \mid \neg\varphi^* \mid \varphi^* \wedge \varphi^* \mid \varphi$$

where $\mathbf{a} \in \mathbf{AP}$ is an atomic proposition, $x \in \mathbb{X}$ is a *variable*, $\gamma \in (0, 1)$ is a discount factor, $\circ$ is the *next* operator, $\text{opt} \in \{\max, \min\}$ is an optimization direction, and λ is the *fixed point operator*.

Intuitively, the *next* operator $\circ^{\text{opt}}$ carries a discount factor γ , which has the effect of discounting the value of what happens in the future time steps. Notice that we introduced two modalities: $\gamma \cdot \circ^{\text{opt}}\varphi$ and $(1 - \gamma) + \gamma \cdot \circ^{\text{opt}}\varphi$. We need the latter as an interpretation of the negation of the former. The optimization direction opt allows to specify whether the value of the formula occurring after the next operator should be maximized or minimized. *When omitted, the default value is max.*

Note that we split the grammar into sub-formulae φ and φ^* . In every formula, φ^* always appears *after* the occurrence of a next operator $\gamma \cdot \circ$. Intuitively the sole purpose of φ^* is to disclose the variable x . Then, each occurrence of a variable x only occurs after a next operator. This enforces the fixed point operator λx to be *syntactically contractive*: every occurrence of a variable x within the scope of a fixed point λx necessarily appears in the

formula subsequent to a next operator $\gamma \cdot \circ$. In consequence, $\lambda x.\varphi$ yields a contraction mapping which intuitively maps every value function V to the values of the formula φ where all (discounted) occurrence of the variable x are replaced by V . Then, the value of the formula φ is the *unique fixed point*¹⁴ of this mapping. A formula φ is *closed* if all occurrences of a variable $x \in \mathbb{X}$ are in the scope of a fixed point operator λx .

Semantics. In the following, we consider value functions $V \in \mathcal{V}$, where $\mathcal{V}$ denotes the set of value functions $V : \mathcal{S} \rightarrow [0, 1]$. A *variable valuation* is a function $\mathcal{E} : \mathbb{X} \rightarrow \mathcal{V}$ which assigns each variable to a value function in $\mathcal{V}$. Notably, we write $\mathcal{E}[x \mapsto V]$ for the valuation obtained by fixing the valuation of the variable $x \in \mathbb{X}$ to $V \in \mathcal{V}$. We also write $V_{\mathcal{E}}(\cdot, \varphi)$ for the value of the formula φ under the particular valuation $\mathcal{E}$, and $V_{\mathcal{E}}^{\pi}(\cdot, \varphi)$ for the related value function in the Markov chain $\mathcal{M}_{\pi}$ induced by the policy π . With MCs, note that the action space is a singleton, so the opt parameter is omitted (min and max vanish, as usual with MCs).

The semantics of VLC is defined by the following rules: for all states $s \in \mathcal{S}$,

$$\begin{aligned}
 V_{\mathcal{E}}(s, \mathbf{a}) &= \mathbb{1}\{\mathbf{a} \in \ell(s)\} \\
 V_{\mathcal{E}}(s, x) &= \mathcal{E}(x)(s) \\
 V_{\mathcal{E}}(s, \neg\varphi) &= 1 - V_{\mathcal{E}}(s, \varphi) \\
 V_{\mathcal{E}}(s, \varphi_1 \wedge \varphi_2) &= \min\{V_{\mathcal{E}}(s, \varphi_1), V_{\mathcal{E}}(s, \varphi_2)\} \\
 V_{\mathcal{E}}(s, \gamma \cdot \circ^{\text{opt}}\varphi) &= \gamma \cdot \text{opt}_{a \in \mathcal{A}} \mathbb{E}_{s' \sim \mathbf{P}(\cdot|s, a)} V_{\mathcal{E}}(s', \varphi) \\
 V_{\mathcal{E}}(s, (1 - \gamma) + \gamma \cdot \circ^{\text{opt}}\varphi) &= (1 - \gamma) + \gamma \cdot \text{opt}_{a \in \mathcal{A}} \mathbb{E}_{s' \sim \mathbf{P}(\cdot|s, a)} V_{\mathcal{E}}(s', \varphi)
 \end{aligned}$$

$$V_{\mathcal{E}}(s, \lambda x.\varphi) \quad \text{is the unique solution of} \quad V^*(s) = V_{\mathcal{E}[x \mapsto V^*]}(s, \varphi) \quad \text{with } V^* \in \mathcal{V}.$$

The value of sub-formulae φ^* is defined similarly. When it is clear from the context, we omit the valuation $\mathcal{E}$ and simply write $V(\cdot, \varphi)$. In particular, when the formula is given in closed form, all the valuations of variables x are fixed in the scope of λx , so there is no need to specify the valuation $\mathcal{E}$.

Negation. While the value of the formula $\gamma \cdot \circ\varphi$ is clearly the definition of the one-step discount of the value of φ , the reason behind the introduction of the formula $(1 - \gamma) + \gamma \cdot \circ\varphi$ is made clear when looking at the negation of the former:

$$\begin{aligned}
 &V(s, \neg(\gamma \cdot \circ^{\max}\varphi)) \\
 &= 1 - V(s, \gamma \cdot \circ^{\max}\varphi)
 \end{aligned}$$

¹⁴the uniqueness is guaranteed by Banach's fixed point theorem.

$$\begin{aligned}
 &= 1 - \gamma \cdot \max_{a \in \mathcal{A}} \mathbb{E}_{s' \sim \mathbf{P}(\cdot|s,a)} V(s', \varphi) \\
 &= 1 - \gamma + \gamma - \gamma \cdot \max_{a \in \mathcal{A}} \mathbb{E}_{s' \sim \mathbf{P}(\cdot|s,a)} V(s', \varphi) \\
 &= (1 - \gamma) + \gamma \left(1 - \max_{a \in \mathcal{A}} \mathbb{E}_{s' \sim \mathbf{P}(\cdot|s,a)} V(s', \varphi) \right) \\
 &= (1 - \gamma) + \gamma \left(1 - \max_{a \in \mathcal{A}} \mathbb{E}_{s' \sim \mathbf{P}(\cdot|s,a)} [1 - V(s', \neg\varphi)] \right) \\
 &= (1 - \gamma) + \gamma \cdot \min_{a \in \mathcal{A}} \mathbb{E}_{s' \sim \mathbf{P}(\cdot|s,a)} V(s', \neg\varphi) \\
 &= V(s, (1 - \gamma) + \gamma \cdot \circ^{\min} \neg\varphi).
 \end{aligned} \tag{2.30}$$

The disjunction of two formulae is obtained via $\varphi_1 \vee \varphi_2 \equiv \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, with

$$\begin{aligned}
 &V(\cdot, \varphi_1 \vee \varphi_2) \\
 &= V(\cdot, \neg(\neg\varphi_1 \wedge \neg\varphi_2)) \\
 &= 1 - V(\cdot, (\neg\varphi_1 \wedge \neg\varphi_2)) \\
 &= 1 - \min\{V(\cdot, \neg\varphi_1), V(\cdot, \neg\varphi_2)\} \\
 &= 1 - \min\{1 - V(\cdot, \varphi_1), 1 - V(\cdot, \varphi_2)\} \\
 &= \max\{V(\cdot, \varphi_1), V(\cdot, \varphi_2)\}.
 \end{aligned} \tag{2.31}$$

Finally, note that $\neg(\lambda x. \varphi) \equiv \lambda x. \neg\varphi[x/\neg x]$ by definition, where $\varphi[x/\neg x]$ denotes the formula obtained by replacing all occurrences of x in φ by $\neg x$.

Remark 11: Linear time vs. branching time properties

Note that VLC allows not only to express linear temporal properties but also *branching time properties*. For instance, consider the formula $\gamma \cdot \circ(\mathbf{a} \wedge \mathbf{b})$. When γ goes to one, the value of the formula is the maximum probability that the agent visits $\mathbf{a}$ and $\mathbf{b}$ in the next time step, i.e., $\lim_{\gamma \rightarrow 1} V(s, \gamma \cdot \circ(\mathbf{a} \wedge \mathbf{b})) = \sup_{\pi} \mathbb{P}_{\pi}(\{\tau \mid \tau \models \circ(\mathbf{a} \wedge \mathbf{b})\}) = \sup_{\pi} \mathbb{P}_{\pi}(\{\tau \mid \tau \models \circ\mathbf{a} \wedge \circ\mathbf{b}\})$. In contrast, the VLC formula $(\gamma \cdot \circ\mathbf{a}) \wedge (\gamma \cdot \circ\mathbf{b})$ expresses the minimum of the maximum probabilities that the agent visits $\mathbf{a}$ in the next time step and the one that the agent visits $\mathbf{b}$, i.e., $\lim_{\gamma \rightarrow 1} V(s, (\gamma \cdot \circ\mathbf{a}) \wedge (\gamma \cdot \circ\mathbf{b})) = \min_{p \in \{\mathbf{a}, \mathbf{b}\}} \sup_{\pi} \mathbb{P}_{\pi}(\{\tau \mid \tau \models \circ p\})$, which is not directly expressible in LTL.

Contraction mappings. Since all formulae are syntactically contractive, all valuations of variables x are contractive as well, and the fixed point operator λx yields a contraction mapping for all free variables x in the scope of the operator. Thus, the fixed points can be approximated through value iteration (see Algorithm 1): given a valuation $\mathcal{E}$, the value

of the formula $\lambda x.\varphi$ is $V_{\mathcal{E}}(\cdot, \lambda x.\varphi) = \lim_{i \rightarrow \infty} V_i$, where $V_0 \in \mathcal{V}$ is arbitrarily¹⁵ initialized then updated via $V_{i+1} = \Lambda_{\varphi} V_i$ through the contraction mapping

$$\Lambda_{\varphi} : \mathcal{V} \rightarrow \mathcal{V} \text{ defined as } \Lambda_{\varphi} V_i = V_{\mathcal{E}[\lambda x \mapsto V_i]}(\cdot, \varphi).$$

3.6 Linear Properties in VLC

Until now, we have seen that ω -regular objectives — and LTL in particular — allow specifying properties that the agent should satisfy during its interaction with the environment. When the latter is unknown, the satisfaction of such properties cannot be learned, meaning that after a finite number of interaction steps, one cannot decide whether or not the agent is about to satisfy a given specification. VLC, on the other hand, provides a way to compute discounted properties through contraction mappings. In this section, we show that VLC can be used to express linear time properties. In particular, this allows expressing ω -regular properties, but with a discounted interpretation. We further prove that this discounted interpretation is PAC learnable, justifying the usage of VLC for the specification of properties in the context of RL.

The results presented in the following hold for $\text{opt} = \max$, but also for $\text{opt} = \min$ by symmetry. Note that when verifying the satisfaction of a formula according to a given policy π , the optimization direction opt is irrelevant as we consider the values of the induced MC $\mathcal{M}_{\pi}$, where $\min$ and $\max$ vanish.

Reachability. The reachability property, denoted as $\Diamond a$ in LTL, requires the agent to *eventually* visit a state labeled with $a \in \mathbf{AP}$. Along trajectories, the property is satisfied as soon as a state labeled with a is visited. In VLC, a reachability specification expresses that the agent needs to reach a state labeled with a as soon as possible. Formally, along a trajectory $\tau = \langle s_0 : \infty, a_0 : \infty \rangle$, the agent receives a “reward” of γ^t if it reaches a state labeled with a , where t is the smallest time step at which a is visited along τ , and 0 if a is never visited. The *discounted reachability property* is expressed as

$$\Diamond^{\gamma} a \equiv \lambda x. (a \vee \gamma \cdot \circ x).$$

The fixed point interpretation of the formula can be intuitively understood as:

“Either the agent is in a state labeled with a , in which case a reward of one is incurred, or the reward of the next time step is discounted”.

The associated contraction mapping, $\Lambda_{\Diamond^{\gamma} a}$, is defined as

$$(\Lambda_{\Diamond^{\gamma} a} V)(s) = \begin{cases} 1 & \text{if } a \in \ell(s), \text{ and} \\ \gamma \cdot \max_{a \in \mathcal{A}} \mathbb{E}_{s' \sim P(\cdot | s, a)} V(s') & \text{otherwise;} \end{cases}$$

¹⁵e.g., from below with $V_0(s) = 0$ or from above with $V_0(s) = 1$ for all $s \in S$.

which yields $V(\cdot, \Diamond^\gamma \mathbf{a}) = \lim_{i \rightarrow \infty} V_i$ through value iteration, with $V_{i+1} = \Lambda_{\Diamond^\gamma \mathbf{a}} V_i$ for all $i \geq 0$. The sequence converges geometrically in γ . Thus, the error can be controlled through the number of iterations i , with $V_i(s) \geq V(s, \Diamond^\gamma \mathbf{a}) - \varepsilon$ so that $\varepsilon = \gamma^i$ when the value function is initialized from below. Notice that one can further implement this value function through the computation of the expected return in a modified MDP $\mathcal{M}^{\Diamond \mathbf{a}}$, where all states labeled with $\mathbf{a}$ are made absorbing, and the agent receives a return of one upon reaching such states, via the reward function $\mathcal{R}(s, a) = \mathbb{1}\{\mathbf{a} \in \ell(s)\} \cdot (1 - \gamma)$ for any $s \in S$ and $a \in \mathcal{A}$. Therefore, optimal deterministic stationary policies exist for $\Diamond^\gamma$ by Theorem 5.

Lemma 5: Reachability in the limit

The value of the formula $\Diamond^\gamma \mathbf{a}$ coincides with the optimal probability of satisfying the associated reachability property in LTL as γ goes to one, i.e.,

$$\lim_{\gamma \rightarrow 1} V(s, \Diamond^\gamma \mathbf{a}) = \sup_{\pi} \mathbb{P}_{\pi}(\{\tau \models \Diamond \mathbf{a}\} \mid s_I = s)$$

for all states $s \in S$.

Remark 12: LTL in the limit is not sufficient for learnability

Lemma 5 already suggests a discounted interpretation of LTL through reachability properties expressed in VLC (an “LTL in the limit” version of the standard logic). Indeed, reachability is all we need to verify LTL properties: all formulae can be translated into a DRA, and their satisfaction can be verified through reachability properties to accepting regions in the product of the MDP with the resulting DRA.

While computing the discounted reachability to such regions suffices to guarantee that the resulting values match the optimal probabilities of satisfying the LTL formulae when the discount goes to one, the detection of accepting regions is, once again, not learnable. To see why, recall that detecting accepting regions boils down to determining the regions of the system where the properties $\Diamond \Box \neg L$ and $\Box \Diamond K$ are witnessed with probability one, for some propositions L and K . The simple fact that (undiscounted) reachability properties are not learnable (cf. Theorem 12) implies that we cannot learn to detect such regions.

Safety. In classic LTL, the safety property $\Box \mathbf{a}$ requires the agent to *always* visit states labeled with $\mathbf{a}$, and is defined as the dual of the reachability property, i.e., $\Box \mathbf{a} \equiv \neg \Diamond \neg \mathbf{a}$. Importantly, the probability of satisfying the safety property is the complement of the probability of the reachability property: $\sup_{\pi} \mathbb{P}_{\pi}(\{\tau \models \Box \mathbf{a}\}) = 1 - \inf_{\pi} \mathbb{P}_{\pi}(\{\tau \models \Diamond \neg \mathbf{a}\})$. In VLC, a safety specification expresses that the agent should

stay in a state labeled with $\mathbf{a}$ as long as possible, resulting in a return of $1 - \gamma^t$ when $\mathbf{a}$ is left (for the first time) at time step t , and 1 if $\mathbf{a}$ is never left.

Precisely, the *discounted safety property* is

$$\Box^\gamma \mathbf{a} \equiv \lambda x. (\mathbf{a} \wedge ((1 - \gamma) + \gamma \cdot \circ x)).$$

Notice that this derivation of the safety property is obtained by the definition of the negation of the discounted reachability property with

$$\Box^\gamma \mathbf{a} \equiv \neg \Diamond_{\min}^\gamma \neg \mathbf{a} \equiv \neg (\lambda x. (\neg \mathbf{a} \vee \gamma \cdot \circ^{\min} x)), \quad (\text{cf. Equations 2.30 and 2.31})$$

where $\Diamond_{\min}^\gamma$ denotes the discounted reachability property with $\text{opt} = \min$. Beyond the negation of the discounted reachability property, the fixed point interpretation of the formula can be intuitively understood as:

“While the agent visits states labeled with $\mathbf{a}$, it receives a reward of $(1 - \gamma)$ and the reward of the next time step is discounted”.

The associated contraction mapping, $\Lambda_{\Box^\gamma \mathbf{a}}$, is defined as

$$(\Lambda_{\Box^\gamma \mathbf{a}} V)(s) = \begin{cases} (1 - \gamma) + \gamma \cdot \max_{a \in \mathcal{A}} \mathbb{E}_{s' \sim P(\cdot | s, a)} V(s') & \text{if } \mathbf{a} \in \ell(s), \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

This yields $V(\cdot, \Box^\gamma \mathbf{a}) = \lim_{i \rightarrow \infty} V_i$ through value iteration, with $V_{i+1} = \Lambda_{\Box^\gamma \mathbf{a}} V_i$ for all $i \geq 0$. Again, the sequence converges geometrically in γ , from above when $V_0(s) = 1$ and from below when $V_0(s) = 0$ for all $s \in \mathcal{S}$. In fact, similarly to the reachability case, one can see this value function as the value of the expected return of a modified MDP $\mathcal{M}^{\circ \neg \mathbf{a}}$, where all states that *are not* labeled with $\mathbf{a}$ are made absorbing, and where a reward of $(1 - \gamma)$ is incurred at each time step while it is possible to avoid such absorbing states, via the reward function $\mathcal{R}(s, a) = \mathbb{1}\{\mathbf{a} \in \ell(s)\} \cdot (1 - \gamma)$ for any $s \in \mathcal{S}$ and $a \in \mathcal{A}$. Once again, optimal deterministic stationary policies exist by Theorem 5.

Lemma 6: Safety in the limit

The value of the formula $\Box^\gamma \mathbf{a}$ coincides with the optimal probability of satisfying the associated safety property in LTL as γ goes to one, i.e.,

$$\lim_{\gamma \rightarrow 1} V(s, \Box^\gamma \mathbf{a}) = \sup_{\pi} \mathbb{P}_{\pi}(\{\tau \mid \tau \models \Box \mathbf{a}\} \mid s_I = s)$$

for all states $s \in \mathcal{S}$.

Persistence and liveness. Recall that (undiscounted) ω -regular properties can be verified in MDPs by checking the reachability probabilities to accepting regions in the product of the MDP with the DRA encoding the given property. The acceptance condition of a DRA involves the satisfaction of properties of the form $\Diamond\Box\neg L$ (*persistence; something good eventually holds forever*) and $\Box\Diamond K$ (*liveness; something good keeps happening*) for some propositions L and K (cf. Definition 20): states labeled with L should be visited only finitely often while those labeled with K should be visited infinitely often. Interestingly, fixed points in VLC can be used to express several alternatives and discounted interpretations of persistence and liveness properties, *through the same formula*:

$$\lambda y. \lambda x. ((\neg a \wedge \alpha \cdot \circ x) \vee (a \wedge ((1 - \beta) + \beta \cdot \circ y)))$$

for discount factors $\alpha, \beta \in (0, 1)$. When β approaches one, the formula requires that the distance between the repeated visits to states labeled with a are as small as possible; $\beta \rightarrow 1$ ensures that a is continually visited while α ensures that this will eventually happen. In contrast, when α approaches one, the formula requires that the number of visits to states labeled with a is as large as possible but arbitrarily spaced; $\alpha \rightarrow 1$ ensures that a will be eventually visited, and β that this remains always the case.

The resulting discounted interpretation of persistence and liveness properties introduces a trade-off between the frequency and the regularity of visits to states labeled with a , leading to the following remarkable observations. First, when $\alpha = \beta$, the discounted interpretation of the formula does not allow to distinguish between persistence and liveness properties. Second, due to the uniqueness of the solution yielded by the value of the formula, the order of the fixed point operator is not important. However, due to Lemmae 5 and 6, the limit of the discount factors are in contrast not interchangeable, as stated in the following lemma.

Lemma 7: Persistence and liveness in the limit [de Alfaro et al., 2003]

The value of the formula $\varphi(\alpha, \beta) = \lambda y. \lambda x. ((\neg a \wedge \alpha \cdot \circ x) \vee (a \wedge ((1 - \beta) + \beta \cdot \circ y)))$ coincides with the optimal probability of satisfying the persistence or liveness properties in LTL as α and β go to one *according to the order in which their limit are taken*, i.e.,

$$\begin{aligned} \lim_{\alpha \rightarrow 1} \lim_{\beta \rightarrow 1} V(s, \varphi(\alpha, \beta)) &= \sup_{\pi} \mathbb{P}_{\pi}(\{\tau \models \Diamond\Box a\} \mid s_I = s), \\ \lim_{\beta \rightarrow 1} \lim_{\alpha \rightarrow 1} V(s, \varphi(\alpha, \beta)) &= \sup_{\pi} \mathbb{P}_{\pi}(\{\tau \models \Box\Diamond a\} \mid s_I = s) \end{aligned}$$

for all states $s \in S$.

In the following, instead of considering specific ω -regular acceptance conditions (such as the Rabin acceptance studied in Section 3.3), we consider a VLC semantics over the

space of infinite trajectories of the MDP where the different interpretations induced by the choices of discount factors can be explored.

Linear λ -calculus: Semantics over Linear Traces

We call the upcoming semantics of VLC over infinite trajectories of $\mathcal{M}$ the *linear λ -calculus* (LLC). Intuitively, LLC assigns each trajectory of $\mathcal{M}$ to a value in $[0, 1]$. We define LLC by considering the semantics of VLC formulae in the augmented infinite Markov chain $\mathcal{M}^{Traj}$ which encodes all the possible infinite trajectories of $\mathcal{M}$ in its state space. Importantly, although the state space of this model is clearly uncountable, this poses no issue: we do not perform any computation within $\mathcal{M}^{Traj}$. Instead, we leverage this model solely as a tool to establish the semantics of VLC over infinite trajectories.

The augmented MC $\mathcal{M}^{Traj} = \langle S^{Traj}, \mathbf{P}^{Traj}, \mathbf{AP}, \ell^{Traj} \rangle$ is defined as follows.¹⁶ The state space is the set of all infinite trajectories of $\mathcal{M}$ augmented with a step counter, i.e.,

$$S^{Traj} = Traj_{\mathcal{M}} \times \mathbb{N} = \{ \langle \tau, t \rangle : \tau = \langle s_{0:\infty}, a_{0:\infty} \rangle \in Traj_{\mathcal{M}} \text{ and } t \geq 0 \}.$$

Then, each state has the form $\langle \tau, t \rangle$, where t represents the current time step t at which the agent is along the trajectory $\tau = \langle s_{0:\infty}, a_{0:\infty} \rangle$, indicating that the agent is in state s_t . From any state $\langle \tau, t \rangle$, the transition function of $\mathcal{M}^{Traj}$ deterministically makes the system transition to the next state along the trajectory $\langle \tau, t+1 \rangle$. Formally, the transition function is defined as $\mathbf{P}^{Traj}(\langle \tau, t+1 \rangle \mid \langle \tau, t \rangle) = 1$ for all $\langle \tau, t \rangle \in S^{Traj}$. Finally, the labeling function ℓ^{Traj} returns the label of the t^{th} state encountered along τ , i.e., $\ell^{Traj}(\langle \tau, t \rangle) = \ell(s_t)$.

Denote by $V_{\mathcal{E}}^{\mathcal{M}^{Traj}}(\cdot, \varphi)$ the value of the VLC formula φ in the infinite MC $\mathcal{M}^{Traj}$ for some arbitrary valuation $\mathcal{E}$. LLC is defined through the objective function $\mathbb{O}^{\varphi} : Traj_{\mathcal{M}} \rightarrow [0, 1]$ which maps any trajectory τ to the value of the formula φ in the infinite MC $\mathcal{M}^{Traj}$, i.e., $\mathbb{O}^{\varphi}(\tau) = V_{\mathcal{E}}^{\mathcal{M}^{Traj}}(\langle \tau, 0 \rangle, \varphi)$.

LLC is *often* linked to the VLC semantics. For example, take the VLC formula $\Diamond^{\gamma} \mathbf{a}$. In LLC, $\mathbb{O}^{\Diamond^{\gamma} \mathbf{a}}(\langle s_{0:\infty}, a_{0:\infty} \rangle) = \gamma^t$, where t is the first time step where $\mathbf{a}$ is visited along the trajectory, with $\mathbf{a} \in \ell(s_t)$. Hence, under any policy π , the expected value of $\mathbb{O}^{\Diamond^{\gamma} \mathbf{a}}$ coincides with the value of the formula, $V^{\pi}(s_0, \Diamond^{\gamma} \mathbf{a})$. The same arguments hold for $\Box^{\gamma} \mathbf{a}$, but do not hold for all formulae: take the formula $\varphi = (\gamma \cdot \circ \mathbf{a}) \wedge (\gamma \cdot \circ \mathbf{b})$. The linear interpretation of φ is not the same as in VLC (see Remark 11). Then, to establish a connection between general linear time properties (carried through LLC) and the semantics of VLC, we need to find the right subset of VLC for which the semantics coincide.

In the classical μ -calculus, Emerson and Jutla [1991] showed that *strongly guarded formulae are sufficient to express all ω -regular properties*.

¹⁶We omit the action space, the reward function, the discount factor, and the initial state from the definition since they are irrelevant for defining LLC.

Definition 27: Strongly guarded formulae

A VLC formula φ is *strongly guarded* if and only if φ is in closed form and consists of a string of fixed point operators followed by a λ -free part ψ , where all the conjunctions have at least one constant argument (i.e., all conjunctive sub-formulae of ψ have the form $\mathbf{a} \wedge \psi'$ for some $\mathbf{a} \in \mathbf{AP}$).

Precisely, φ is generated by the following grammar:

$$\begin{aligned}\varphi &:= \lambda x. \varphi \mid \psi \\ \psi &:= \mathbf{a} \mid \neg \mathbf{a} \mid \psi \vee \psi \mid \mathbf{a} \wedge \psi \mid \neg \mathbf{a} \wedge \psi \mid \gamma \cdot \circ^{\text{opt}} \chi \mid (1 - \gamma) + \gamma \cdot \circ^{\text{opt}} \chi \\ \chi &:= x \mid \chi \vee \chi\end{aligned}$$

where $\mathbf{a} \in \mathbf{AP}$, $x \in \mathbb{X}$, and opt is always equal to $\max$ (or always $\min$, in the symmetric case). Furthermore, we impose φ to be closed.

Remarkably, for any strongly guarded formula φ , LLC coincides with the semantics of VLC, as stated in the following theorem.

Theorem 13: Linear interpretation of VLC [de Alfaro et al., 2003]

For any strongly guarded formula φ and any state $s \in S$, the value of φ coincides with the expected value of the linear objective $\mathbb{O}^\varphi$, i.e.,

$$V(s, \varphi) = \sup_{\pi} \mathbb{E}_{\pi}^{\mathcal{M}} [\mathbb{O}^\varphi \mid s_I = s].$$

This theorem is a key result since it establishes that a discounted interpretation of ω -regular properties, formalized as LLC, can be verified through the fixed point semantics of VLC.

3.7 VLC's Linear Properties are PAC Learnable

We now establish the learnability of the linear properties expressed in VLC. This is very important: it means that the discounted interpretation of ω -regular properties, formalized as LLC, can be learned from data, which justifies the usage of VLC as a formalism for the specification of properties in the context of RL.

Theorem 14: LLC is PAC learnable *

Given any policy π and any *closed* VLC formula φ , there exists a learning algorithm $\mathcal{L}_{\mathcal{M}}^{\pi}$ which stops after a finite number of iterations $T \geq 0$ and outputs a dataset $\mathcal{D}$ from which one can construct an estimator $\mathcal{E}_{\mathcal{D}} \in \Delta(\text{Traj}_{\mathcal{M}})$ where

$$\left| \mathbb{E}_{\pi}^{\mathcal{M}} [\mathbb{O}^{\varphi} \mid s_I = s] - \mathbb{E}_{\tau \sim \mathcal{E}_{\mathcal{D}}} [\mathbb{O}^{\varphi}] \right| \leq \varepsilon$$

with probability at least $1 - \delta$ for all states $s \in \mathcal{S}$, where $\varepsilon, \delta \in (0, 1)$ are the error and confidence parameters of the learning algorithm, respectively.

Proof. The proof follows from the fact that $\mathbb{O}^{\varphi}$ is uniformly continuous. Precisely, we consider the *Cantor metric* d_C over the trajectory space $\text{Traj}_{\mathcal{M}}$, defined as $d_C(\tau, \tau') = \frac{1}{2^t}$, where t is the length of the maximal prefix common between the two trajectories $\tau, \tau' \in \text{Traj}_{\mathcal{M}}$. Then, for every $\varepsilon > 0$, there is a $\kappa > 0$ such that for all $\tau, \tau' \in \text{Traj}_{\mathcal{M}}$,

$$d_C(\tau, \tau') < \kappa \text{ implies } \left| \mathbb{O}^{\varphi}(\tau) - \mathbb{O}^{\varphi}(\tau') \right| < \varepsilon.$$

This stems from the syntactic contractivity of φ : each application of the next operator $\gamma \cdot \circ$ or $(1 - \gamma) + \gamma \cdot \circ$ discounts the subsequent payoff in the trajectory.

Formally, let $0 < \gamma < 1$ be the largest discount occurring in the formula φ and $H \geq 0$ be an arbitrary horizon. Then, for all trajectories $\tau = \langle s_0 : \infty, a_0 : \infty \rangle, \tau' = \langle s'_0 : \infty, a'_0 : \infty \rangle \in \text{Traj}_{\mathcal{M}}$ sharing the same prefix of length H , meaning $s_t = s'_t, a_t = a'_t$ for all $t \leq H$, we have that the payoff accumulated from time step H onward is bounded by γ^H . Therefore, we have $|\mathbb{O}^{\varphi}(\tau) - \mathbb{O}^{\varphi}(\tau')| \leq \gamma^H$. For any $\varepsilon > 0$, we have that $\gamma^H < \varepsilon$ if and only if $H > \frac{\log \varepsilon}{\log \gamma}$. Then, taking $\kappa < 2^{\frac{-\log \varepsilon}{\log \gamma}}$ yields the result.

The PAC learnability finally follows from the result of Yang et al. [2023a, Theorem 1]: *every uniformly continuous function is PAC learnable.* $\square$

3.8 Model Checking ω -regular Properties through VLC

Theorem 13 opens the door to the verification of ω -regular properties through the fixed point semantics of VLC, by encoding ω -regular acceptance conditions through VLC's strongly guarded formulae. Recall that the verification of ω -regular properties in MDPs may be performed through the construction of a product with a deterministic Rabin automaton $\mathcal{A}$. The acceptance condition of a Rabin automaton consists of n pairs of the form $\langle L_i, K_i \rangle$ for $0 \leq i \leq n - 1$, where L_i and K_i are subsets of states of the automaton $\mathcal{A}$: for at least one pair, L_i must be visited only finitely often, while K_i must be visited infinitely often.

Rabin chain. In the product MDP $\mathcal{M}^{\mathcal{A}}$, each pair is encoded through the labeling function of $\mathcal{M}^{\mathcal{A}}$, and states of the product that encodes L_i or K_i are re-labeled with the corresponding atomic proposition (see Definition 21). This way, $2n$ atomic propositions are added to $\mathbf{AP}$. Precisely, for all $0 \leq i \leq n-1$, we add the propositions $\mathbf{a}_{2i}$ and $\mathbf{a}_{2i+1}$, respectively encoding the sets K_i and L_i . Then, the acceptance condition of the Rabin automaton is encoded in LTL through the formula

$$\bigvee_{i=0}^{n-1} (\Box \Diamond \mathbf{a}_{2i} \wedge \neg \Box \Diamond \mathbf{a}_{2i+1}).$$

In the following, we consider particular DRAs where the acceptance condition forms a chain

$$\emptyset = K_n \subseteq L_n \subseteq K_{n-1} \subseteq L_{n-1} \subseteq \dots \subseteq K_1 \subseteq L_1 \subseteq K_0 = Q,$$

where Q is the state space of the DRA $\mathcal{A}$. Let $S^{\mathcal{A}} = S \times Q$ be the state space of the product MDP $\mathcal{M}^{\mathcal{A}}$ and denote by $S_{\mathbf{a}}^{\mathcal{A}}$ the set of states labeled with $\mathbf{a} \in \mathbf{AP}$, i.e., $S_{\mathbf{a}}^{\mathcal{A}} = \{\langle s, q \rangle \in S^{\mathcal{A}} \mid \mathbf{a} \in \ell^{\mathcal{A}}(s, q)\}$. Then, we have

$$\emptyset = S_{\mathbf{a}_{2n}}^{\mathcal{A}} \subseteq S_{\mathbf{a}_{2n-1}}^{\mathcal{A}} \subseteq \dots \subseteq S_{\mathbf{a}_1}^{\mathcal{A}} \subseteq S_{\mathbf{a}_0}^{\mathcal{A}} = S^{\mathcal{A}}.$$

Such a chain of inclusions of Rabin acceptance sets is called a *Rabin chain*.

In fact, for every DRA, there exists another equivalent DRA with Rabin chain acceptance condition [Krishnan et al., 1995].

Parity condition. An alternative interpretation of the Rabin chain acceptance condition is to assign priorities to the states of the product MDP based on their position in the chain. Precisely, for all $0 \leq i \leq 2n-1$, we assign to the state $\langle s, q \rangle \in S^{\mathcal{A}}$ the *priority* i via a new atomic proposition $\mathbf{p}_i$ so that

$$\mathbf{p}_i \in \ell^{\mathcal{A}}(s, q) \text{ if and only if } \langle s, q \rangle \in S_{\mathbf{a}_i}^{\mathcal{A}} \setminus S_{\mathbf{a}_{i+1}}^{\mathcal{A}}.$$

Recall that for any trajectory $\tau \in \text{Traj}_{\mathcal{M}^{\mathcal{A}}}$, $\text{inf}(\tau)$ is the set of states visited infinitely often along τ (cf. Section 1.4, Equation 2.4). Then, let

$$\text{MaxPriority}(\tau) = \max \left\{ 0 \leq i \leq 2n-1 \mid S_{\mathbf{p}_i}^{\mathcal{A}} \cap \text{inf}(\tau) \neq \emptyset \right\}$$

be the maximal priority seen infinitely often along τ . If $\text{MaxPriority}(\tau)$ is odd, no acceptance pair can be satisfied. Precisely, this implies the existence of a subset L_i that occurs infinitely often, but should only be visited a finite number of times. Moreover, this subset is included in all sets $L_{j < i}$, each of which should also be visited only finitely

often. As this subset is the smallest one¹⁷ of the chain visited infinitely often, there are no smaller subsets $K_{j>i}$ occurring infinitely often, thus the acceptance condition cannot be fulfilled.

On the other hand, if $\text{MaxPriority}(\tau)$ is even, the acceptance condition is satisfied: that means there exists a pair $\langle L_i, K_i \rangle$ where $L_i \subseteq K_i$ is visited only finitely often (otherwise the maximum priority would have been odd), while $K_i \setminus L_i$ is visited infinitely often.

Therefore, the acceptance condition of a DRA whose acceptance condition forms a Rabin chain can be expressed as the *parity condition*

$\text{MaxPriority}(\tau)$ is even.

In fact, DRAs with acceptance conditions forming a Rabin chain can be viewed as *deterministic parity automata*. Any LTL property can be encoded through deterministic parity automata of state space doubly exponential in the size of the formula [Esparza et al., 2017] (compare with Theorem 11), and this construction is also part of the RABINIZER tool [Kretínský et al., 2018].

Model checking. The parity condition can be encoded through the following VLC's strongly guarded formula [Emerson and Jutla, 1991; de Alfaro et al., 2001, 2003]:

$$\varphi_{\text{Parity}}(y_0, \dots, y_{2n-1}) = \lambda x_{2n-1} \dots \lambda x_0. \bigvee_{i=0}^{2n-1} (\mathbf{p}_i \wedge \bullet_i x_i)$$

where $\bullet_i \equiv y_i \cdot \circ$ whenever i is odd, and $\bullet_i \equiv (1 - y_i) + y_i \cdot \circ$ if i is even with discounts $y_i \in (0, 1)$ for all $0 \leq i \leq 2n - 1$. We simply write φ_{Parity} when it is clear from the context. As for persistence and liveness properties, the chain of discount factors represents the different possible tradeoffs for expressing the parity condition. Importantly, the value of the parity objective $V(\cdot, \varphi_{\text{Parity}})$ can be approximated via VI. In particular, the nested fixed points can be computed by using one value function $V^i \in \mathcal{V}$ per priority.

Precisely, let $i \in \{0, \dots, 2n - 1\}$ be a priority and let $\mathcal{E}$ be some valuation for the variables $x_0, \dots, x_{2n-1}$. The value of φ_{Parity} for the i^{th} priority is the unique fixed point of

$$\begin{aligned} V^{i*}(s, q) &= V_{\mathcal{E}[x_j \mapsto V^{j*} : 0 \leq j \leq 2n-1]} \left(\langle s, q \rangle, \bigvee_{j=0}^{2n-1} (\mathbf{p}_j \wedge \bullet_j x_j) \right) \\ &= \max_{0 \leq j \leq 2n-1} \min \left\{ \mathbb{1} \{ \mathbf{p}_j \in \ell(s, q) \}, \right. \\ &\quad \left. \mathbb{1} \{ j \text{ is even} \} \cdot (1 - y_j) + y_j \cdot \max_{a \in \mathcal{A}} \max_{\langle s', q' \rangle \sim \mathbf{P}^A(\cdot | s, q, a)} \mathbb{E} V^{j*}(s', q') \right\}. \end{aligned} \tag{2.32}$$

¹⁷the higher is the priority, the smaller is the corresponding subset.

Let us have a closer look at Equation 2.32. First, notice that by definition, a unique priority is assigned to each state. Then, the min in the equation ensures the outer $\max_j$ to select the contraction associated with the priority assigned to the input state $\langle s, q \rangle \in \mathcal{S}^A$. Let $p : \mathcal{S}^A \rightarrow \{0, \dots, 2n-1\}$ be the function assigning each state to its priority $p(s, q) = i$ if and only if $\mathbf{p}_i \in \ell^A(s, q)$. Then, one can rewrite the value function of the i^{th} fixed point as

$$V^{i*}(s, q) = \begin{cases} (1 - \gamma_j) + \gamma_j \cdot \max_{a \in A} \mathbb{E}_{s', q' \sim \mathbf{P}^A(\cdot | s, q, a)} V^{j*}(s', q') & \text{if } j = p(s, q) \text{ is even,} \\ \gamma_j \cdot \max_{a \in A} \mathbb{E}_{s', q' \sim \mathbf{P}^A(\cdot | s, q, a)} V^{j*}(s', q') & \text{if } j = p(s, q) \text{ is odd.} \end{cases}$$

The nested nature of the fixed points in Equation 2.32 makes them difficult to compute directly. Alternatively, de Alfaro et al. [2003] showed that the value of the parity objective can be approximated through a *sequential* (rather than nesting) VI procedure. Assume that we are given a vector $\langle k_0, k_1, \dots, k_{2n-1} \rangle \in \mathbb{N}^{2n-1}$. For each priority, ranging from $i = 0$ to $2n - 1$, the idea is to estimate the values V^i of the i^{th} fixed point through k_i VI steps, resulting in a value function $V_{k_i}^i$. Intuitively, at each VI step $k \geq 0$, V_k^i is updated by leveraging the (fixed) previous priority values $V_{k_j}^j$ for $j < i$. After the k_i^{th} iteration, the vector $V_{k_i}^i$ is in turn frozen, and then the sequential algorithm proceeds to the next priority $i + 1$. Formally, for all $0 \leq i \leq 2n - 1$, let $V_0^i = V_0$ be some initial valuation (from below, V_0 is initialized to zero for all states, and from above, V_0 is initialized to one for all states). Then, for all $0 < k \leq k_i$ and $\langle s, q \rangle \in \mathcal{S}^A$, let $p(s, q) = j$, we have

$$V_k^i(s, q) = \begin{cases} (1 - \gamma_i) \cdot \mathbb{1}\{i \text{ is even}\} + \gamma_i \cdot \max_{a \in A} \mathbb{E}_{s', q' \sim \mathbf{P}^A(\cdot | s, q, a)} V_{k-1}^i(s', q') & \text{if } j = i, \\ V_{k_j}^j(s, q) & \text{if } j < i, \text{ and} \\ V_0(s, q) & \text{otherwise.} \end{cases}$$

We denote the unique fixed point of V_k^i by

$$V^i = \lim_{k_i \rightarrow \infty} \lim_{k_{i-1} \rightarrow \infty} \dots \lim_{k_0 \rightarrow \infty} V_{k_i}^i = \begin{cases} (1 - \gamma_i) \cdot \mathbb{1}\{i \text{ is even}\} + \gamma_i \cdot \max_{a \in A} \mathbb{E}_{\mathbf{P}^A(\cdot | a)} [V^i] & \text{if } j = i, \\ V^j & \text{if } j < i, \text{ and} \\ V_0 & \text{otherwise.} \end{cases}$$

Lemma 8: Sequential VI error for parity objectives

Whatever the initial valuation $V_0 \in \mathcal{V}$, the error in the i^{th} fixed point, $\|V_{k_i}^i - V^i\|_\infty$, is upper bounded by $\varepsilon_i = \gamma_0^{k_0}$ if $i = 0$ and $\varepsilon_i = \frac{\varepsilon_{i-1}}{1 - \gamma_{k_i}} + \gamma^{k_i}$ otherwise.

Proof. We proceed by induction on the priority $i \in \{0, \dots, 2n-1\}$. We start with $i = 0$.

$$\begin{aligned} \|V_{k_0}^0 - V^0\|_\infty = \sup_{s,q} & \left| \left[(1 - \gamma_0) + \gamma_0 \cdot \max_{a \in \mathcal{A}} \mathbb{E}_{s',q' \sim \mathbf{P}^{\mathcal{A}}(\cdot|s,q,a)} V_{k_0-1}^0(s',q') \right. \right. \\ & \left. \left. - (1 - \gamma_0) - \gamma_0 \cdot \max_{a \in \mathcal{A}} \mathbb{E}_{s',q' \sim \mathbf{P}^{\mathcal{A}}(\cdot|s,q,a)} V^0(s',q') \right] \cdot \mathbb{1}\{p(s,q) = 0\} \right. \\ & \left. + (V_0(s,q) - V^0(s,q)) \cdot \mathbb{1}\{p(s,q) > 0\} \right| \end{aligned}$$

Note that for all $i \in \{0, \dots, 2n-1\}$,

$$\begin{aligned} & \gamma_i \cdot \sup_{s,q} \left| \max_{a \in \mathcal{A}} \mathbb{E}_{s',q' \sim \mathbf{P}^{\mathcal{A}}(\cdot|s,q,a)} V_{k_i-1}^i(s',q') - \max_{a \in \mathcal{A}} \mathbb{E}_{s',q' \sim \mathbf{P}^{\mathcal{A}}(\cdot|s,q,a)} V^i(s',q') \right| \\ & \leq \gamma_i \cdot \sup_{s,q} \left| \max_{a \in \mathcal{A}} \left(\mathbb{E}_{s',q' \sim \mathbf{P}^{\mathcal{A}}(\cdot|s,q,a)} V_{k_i-1}^i(s',q') - \mathbb{E}_{s',q' \sim \mathbf{P}^{\mathcal{A}}(\cdot|s,q,a)} V^i(s',q') \right) \right| \\ & \leq \gamma_i \cdot \sup_{s,q} \max_{a \in \mathcal{A}} \mathbb{E}_{s',q' \sim \mathbf{P}^{\mathcal{A}}(\cdot|s,q,a)} \left| V_{k_i-1}^i(s',q') - V^i(s',q') \right| \\ & \leq \gamma_i \cdot \|V_{k_i-1}^i - V^i\|_\infty \end{aligned} \tag{2.33}$$

Then, we have

$$\|V_{k_0}^0 - V^0\|_\infty \leq \gamma_0 \cdot \|V_{k_0-1}^0 - V^0\|_\infty \leq \gamma_0^2 \cdot \|V_{k_0-2}^0 - V^0\|_\infty \leq \dots \leq \gamma_0^{k_0} \underbrace{\|V_0 - V^0\|_\infty}_{\leq 1} \leq \gamma_0^{k_0}.$$

Now, let $0 < j < 2n-1$ and assume that $\|V_{k_j}^j - V^j\|_\infty \leq \varepsilon_j = \frac{\varepsilon_{j-1}}{1-\gamma_{k_j}} + \gamma^{k_j}$. For $i > j$, we have

$$\begin{aligned} & \|V^{k_i} - V^i\|_\infty \\ & = \sup_{s,q} \left[\mathbb{1}\{p(s,q) = i\} \cdot \left| (1 - \gamma_i) \cdot \mathbb{1}\{i \text{ is even}\} + \gamma_i \cdot \max_{a \in \mathcal{A}} \mathbb{E}_{s',q' \sim \mathbf{P}^{\mathcal{A}}(\cdot|s,q,a)} V_{k_i-1}^i(s',q') \right. \right. \\ & \quad \left. \left. - (1 - \gamma_i) \cdot \mathbb{1}\{i \text{ is even}\} - \gamma_i \cdot \max_{a \in \mathcal{A}} \mathbb{E}_{s',q' \sim \mathbf{P}^{\mathcal{A}}(\cdot|s,q,a)} V^i(s',q') \right| \right. \\ & \quad \left. + \mathbb{1}\{p(s,q) = j < i\} \cdot \left| V_{k_j}^j(s,q) - V^j(s,q) \right| \right. \\ & \quad \left. + \mathbb{1}\{p(s,q) > i\} \cdot |V_0(s,q) - V^0(s,q)| \right] \end{aligned}$$

$$\begin{aligned}
 &\leq \sup_{s,q} \left| (1 - \gamma_i) \cdot \mathbb{1}\{i \text{ is even}\} + \gamma_i \cdot \max_{a \in \mathcal{A}} \mathbb{E}_{s', q' \sim \mathbf{P}^{\mathcal{A}}(\cdot | s, q, a)} V_{k_{i-1}}^i(s', q') \right. \\
 &\quad \left. - (1 - \gamma_i) \cdot \mathbb{1}\{i \text{ is even}\} - \gamma_i \cdot \max_{a \in \mathcal{A}} \mathbb{E}_{s', q' \sim \mathbf{P}^{\mathcal{A}}(\cdot | s, q, a)} V^i(s', q') \right| \\
 &\quad + \sup_{j < i} \|V_{k_j}^j - V^j\|_{\infty} \\
 &\leq \gamma_i \cdot \|V_{k_{i-1}}^i - V^i\|_{\infty} + \sup_{j < i} \|V_{k_j}^j - V^j\|_{\infty} \quad (\text{see Equation 2.33}) \\
 &\leq \gamma_i \cdot \|V_{k_{i-1}}^i - V^i\|_{\infty} + \varepsilon_{i-1} \\
 &\leq \varepsilon_{i-1} + \gamma_i \varepsilon_{i-1} + \gamma_i^2 \varepsilon_{i-1} \cdots + \gamma_i^{k_i-1} \varepsilon_{i-1} + \gamma_i^{k_i} \cdot \|V_0^i - V^i\|_{\infty} = \sum_{t=1}^{k_i} \gamma_i^{t-1} \cdot \varepsilon_{i-1} + \gamma_i^{k_i} \\
 &\leq \frac{\varepsilon_{i-1}}{1 - \gamma_i} + \gamma_i^{k_i}.
 \end{aligned}$$

□

Theorem 15: Parity Approximations [de Alfaro et al., 2003]

Let $\varphi_{\text{Parity}}(\gamma_0, \dots, \gamma_{2n-1})$ be the VLC's parity encoding of the acceptance condition of a deterministic Rabin chain automaton $\mathcal{A}$ and $\langle k_0, \dots, k_{2n-1} \rangle$ be a vector of integers specifying the number of VI steps to perform for each fixed points of φ_{Parity} . Then, let $V_{\text{Parity}}^{\perp}$ and $V_{\text{Parity}}^{\top}$ be the values obtained by sequentially approximating each fixed point for k_i steps, respectively from below and above. Then, the following holds:

1. for each state $\langle s, q \rangle \in \mathcal{S}^{\mathcal{A}}$, we have

$$V(\langle s, q \rangle, \varphi_{\text{Parity}}) - \varepsilon_{2n-1} \leq V_{\text{Parity}}^{\perp}(s, q) \leq V_{\text{Parity}}^{\top}(s, q) \leq V(\langle s, q \rangle, \varphi_{\text{Parity}}) + \varepsilon_{2n-1};$$

2. taking the limit of the discount factors to 1 in the right order yields the probability of satisfying the parity objective:

$$\begin{aligned}
 &\lim_{\gamma_{2n-1} \rightarrow 1} \dots \lim_{\gamma_0 \rightarrow 1} V(\langle s, q \rangle, \varphi_{\text{Parity}}(\gamma_0 \dots \gamma_{2n-1})) \\
 &= \sup_{\pi} \mathbb{P}_{\pi}^{\mathcal{M}^{\mathcal{A}}} \left(\{ \tau \in \text{Traj}_{\mathcal{M}^{\mathcal{A}}} \mid \text{MaxPriority}(\tau) \text{ is even} \} \right) \\
 &= \sup_{\pi} \mathbb{P}_{\pi}^{\mathcal{M}} \left(\{ \tau = \langle s_0 : \infty, a_0 : \infty \rangle \in \text{Traj}_{\mathcal{M}} \mid w \models \mathcal{A}, \text{ where } w = \ell(s_0)\ell(s_1)\ell(s_2)\dots \} \right).
 \end{aligned}$$

Discussion. Theorem 15 is the cornerstone of the verification of discounted ω -regular properties via VLC. It provides a way to approximate the value of the parity objective

through both a discounted interpretation of the latter and value iteration. Furthermore, when the discount factors go to one, we retrieve the optimal probability of satisfying the acceptance condition of a deterministic Rabin chain automaton.

This suggests that we do not even need to specify the desired properties through VLC: we can specify the desired agent's behaviors through an LTL formula, translate it into a deterministic Rabin chain automaton (or, equivalently, a deterministic parity automaton), specify a discount factor for each priority, and then use the nested VI procedure to approximate, in the product of the MDP and the automaton, the value of the parity objective. Moreover, the latter is equivalent to the optimal probability of satisfying the original LTL formula when the discounts go to one.

Finally, while Theorem 15 ensures that the values of any discounted ω -regular objective can be approximated with any arbitrary precision, Theorem 14 ensures their learnability since the former can be expressed as a strongly guarded formula.

As the cherry on the cake, we will show in the following section that VLC characterizes the *bisimilarity* between MDPs through a distance function. In the undiscounted setting, *bisimulation* is an equivalence relation that preserves the probability of satisfying all LTL properties. In the discounted setting, a quantitative notion of bisimulation can be defined, and we will show that it is characterized by a distance (pseudo-)metric spanning over all VLC formulae. Notably, while the exact bisimulation is sensitive to small perturbations in transition probabilities, the discounted counterpart is robust to such perturbations and assigns a distance between two models: when close to zero, the values of all the VLC formulae are close in both models, whereas a distance close to one indicates that the values of the formulae are far apart.

4 Bisimulation

Bisimulation serves as a fundamental concept for comparing potential agent behaviors across diverse models. In the context of MDPs, a *stochastic* bisimulation is an equivalence relation that preserves both the probability of fulfilling properties — particularly those expressed through LTL formulae — and the optimal expected return. Hence, two *bisimilar* models satisfy the same LTL properties with the same probabilities and yield identical expected return under their optimal policy. Consequently, if one is able to provide a model bisimilar to the actual environment (potentially relatively smaller), then it can be leveraged to verify the original environment since the two models satisfy the same properties, with equal probabilities.

In essence, to ensure this crucial property, the bisimulation relation is established among the states of the system. This necessitates that states in bisimulation are capable of mimicking each other individual steps. This further means that labels, immediate

rewards, and transition probabilities must be identical for those states. This relation can be lifted to MDPs by requiring their initial state to be in relation.

As with any equivalence relation, bisimulation provides a binary metric between states and models: they are either bisimilar or they are not. Consequently, models that are nearly identical but have slight variations in transition probabilities or rewards might be treated as entirely different. This mirrors an observation we made in LTL (see the discussion about *discounting* in Section 3.5): the satisfaction of linear properties yields a binary outcome, so even minor deviations in trajectories after a certain point can lead to vastly different results. This contrasts with VLC's linear properties, where the discounting effect allows trajectories with nearly identical prefixes to yield nearly identical outcomes. This connection extends bisimulation to the discounted setting through a quantitative measure called the *bisimulation pseudo-metric*, which assigns a real-valued distance between states. A distance closer to zero indicates closer values for both the expected return and all VLC formulae in the compared models.

4.1 The Bisimulation Relation

In the following, we formally introduce the *exact* stochastic bisimulation relation, and we then discuss the related distance function. Along this section, note that we may consider MDPs with general state-action spaces. Since bisimulation is an equivalence relation, we start by defining the notion of *equivalence classes*.

Definition 28: Equivalence classes

Let $\mathcal{R}$ be a *relation* on $\mathcal{S}$. Let $T \subseteq \mathcal{S}$, we denote the set of states in relation with T as

$$\mathcal{R}(T) = \{s' \in \mathcal{S} \mid \text{there is a state } s \in T \text{ so that } s\mathcal{R}s'\}.$$

Then, T is $\mathcal{R}$ -closed if $\mathcal{R}(T) \subseteq T$: it is the largest subset so that $s\mathcal{R}s'$ for all $s, s' \in T$. The set of *equivalence classes* of $\mathcal{R}$, denoted $\mathcal{S}/\mathcal{R}$ is a partition of the space $\mathcal{S}$ according to the relation $\mathcal{R}$ (i.e., the *quotient* of $\mathcal{S}$ w.r.t. $\mathcal{R}$). Formally, it is defined as

$$\mathcal{S}/\mathcal{R} = \{T \in \Sigma(\mathcal{S}) \mid T \text{ is } \mathcal{R}\text{-closed, i.e., } \mathcal{R}(T) \subseteq T\}.$$

Definition 29: Bisimulation

A *stochastic bisimulation* $\mathcal{R}$ is an *equivalence relation* on $\mathcal{S}$ such that for all pairs of states $s_1, s_2 \in \mathcal{S}$, $s_1 \mathcal{R} s_2$ if and only if:

- (i) their labels are the same: $\ell(s_1) = \ell(s_2)$;
- (ii) they share the same immediate rewards: $\mathcal{R}(s_1, a) = \mathcal{R}(s_2, a)$ for all $a \in \mathcal{A}$; and
- (iii) their transition probabilities to equivalence classes of $\mathcal{R}$ are the same:
 $\mathbf{P}(T \mid s_1, a) = \mathbf{P}(T \mid s_2, a)$ for all $a \in \mathcal{A}$ and $T \in \mathcal{S}/\mathcal{R}$.

We denote the largest bisimulation relation by $\sim$.

Given a policy π for $\mathcal{M}$, the largest bisimulation relation associated with the states of $\mathcal{M}_\pi$ is $\sim_\pi$. If π is given as a Mealy machine (Property 1) with initial memory state q_I , we may simply write $s_1 \sim_\pi s_2$ for the relation over the induced Markov chain (Definition 8), i.e., $\langle s_1, q_I \rangle \sim_\pi \langle s_2, q_I \rangle$ when $s_1, s_2 \in \mathcal{S}$ belong to the original state space of $\mathcal{M}$.

Remark 13: Bisimulation, labels, and rewards

According to the context, the requirements (i) and (ii) in Definition 29 can be relaxed. If the goal is solely to reason about rewards and expected return, requirement (i) is unnecessary. Similarly, if the goal is to verify properties through logical specification formalisms, requirement (ii) is irrelevant and can be ignored.

The bisimulation between states of the system can be easily generalized to MDPs: in that case, it suffices to consider the disjoint union of their state spaces. Two models are then equivalent if their respective initial state are both in bisimulation.

Definition 30: Bisimulation between MDPs

Let $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathbf{P}, \mathcal{R}, \mathbf{AP}, \ell, s_I, \gamma \rangle$ and $\mathcal{M}' = \langle \mathcal{S}', \mathcal{A}', \mathbf{P}', \mathcal{R}', \mathbf{AP}', \ell', s'_I, \gamma' \rangle$ be two MDPs. Consider a unified model which embeds the two MDPs, defined as

$$\mathcal{M} \uplus \mathcal{M}' = \langle \mathcal{S} \uplus \mathcal{S}' \cup \{s_\perp\}, \mathcal{A} \cup \mathcal{A}', \mathbf{P}^*, \mathcal{R}^*, \mathbf{AP} \cup \mathbf{AP}', \ell^*, s_\perp^*, \max\{\gamma, \gamma'\} \rangle,$$

where $\uplus$ denotes the *disjoint union* operator, i.e., the union between sets assumed disjoint. $\mathbf{P}^*(\cdot \mid s, a)$ is equal to $\mathbf{P}(\cdot \mid s, a)$ (resp. $\mathbf{P}'(\cdot \mid s, a)$) if $s \in \mathcal{S}$ (resp. $s \in \mathcal{S}'$). If s is a state of $\mathcal{M}$ while a is not an action of $\mathcal{M}$ (or conversely, $s \in \mathcal{S}'$ and $a \notin \mathcal{A}'$), then the system deterministically transitions to a special absorbing state $s_\perp$. $\mathcal{R}^*$ and ℓ^* are defined similarly. $\mathcal{R}^*$ incurs a reward of zero when transitioning to $s_\perp$ and $\ell(s_\perp) = \emptyset$. Finally $s_\perp^*$ is an arbitrary state from either $\mathcal{S}$ or $\mathcal{S}'$.

We say that $\mathcal{M}$ and $\mathcal{M}'$ are in bisimulation, denoted $\mathcal{M} \sim \mathcal{M}'$ if and only if $s_I \sim s'_I$ in $\mathcal{M} \uplus \mathcal{M}'$.

Example 15: Bisimulation

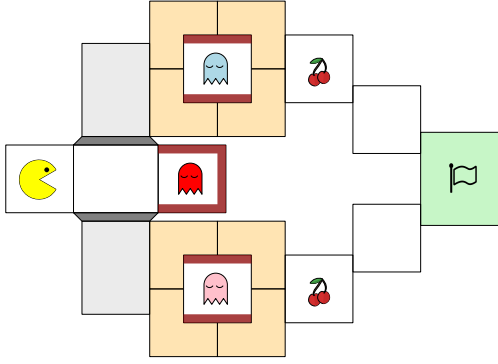
Consider a more complex version of the grid world environment from Example 3, as illustrated in Figure 2.13a. Here, the agent  navigates within a grid world alongside three adversaries: , , and . Similar to the original example, these adversaries are initially asleep. In the first steps, attempting to move upward or downward from the cell adjacent to the sleeping adversary  awakens the latter, prompting it to pursue the agent. When awakened, the agent has a 90% chance of escaping but randomly falls into one of two gray areas above or below. Once in these areas, the agent is unable to return to the white cell and must proceed to the orange zone. Within this orange zone, the agent encounters one of two sleeping adversaries:  or . Moving up, down, or remaining in the same cell poses a 50% risk of awakening the adversary, resulting in the agent being eaten. Moving horizontally allows the agent to navigate along the walls and evade awakening the adversaries. Once the orange area is escaped, the agent may reach the green area, where the goal  is located.

The MDP $\mathcal{M}_1$ depicted in Figure 2.13b represents this environment. For the sake of clarity, the atomic propositions forming the label are depicted this time within each state. As usual, we omitted irrelevant actions leading to self-loops with zero return. We have also omitted the self-loops in states s_7 and s_{17} leading to a reward of -0.1 for clarity. Notice that we only used one atomic proposition  for each state representing an adversary since the outcome is the same in each of these states (being trapped and receiving a reward of -0.1 at each step). Notably, the state space of $\mathcal{M}_1$ is composed of 20 states.

One may already notice from Figure 2.13a, depicting the environment, that some parts of the state space are similar: the agent behaves similarly from the top and the bottom of the Figure. Furthermore, one may also notice that the agent behaves similarly to the gray and the orange areas. We show that those areas are actually bisimilar regions of the state space.

Precisely, the states in bisimulation are the following:

- s_2 and s_{12} (the gray area);
- s_7, s_{10}, s_{17} (representing the adversaries , , and );
- s_3, s_4, s_{13}, s_{14} (from the leftmost orange area);
- s_5, s_6, s_{15}, s_{16} (from the rightmost orange area);
- s_8 and s_{18} (labeled with ); as well as
- s_9 and s_{19} .



(a) A more complex grid world environment

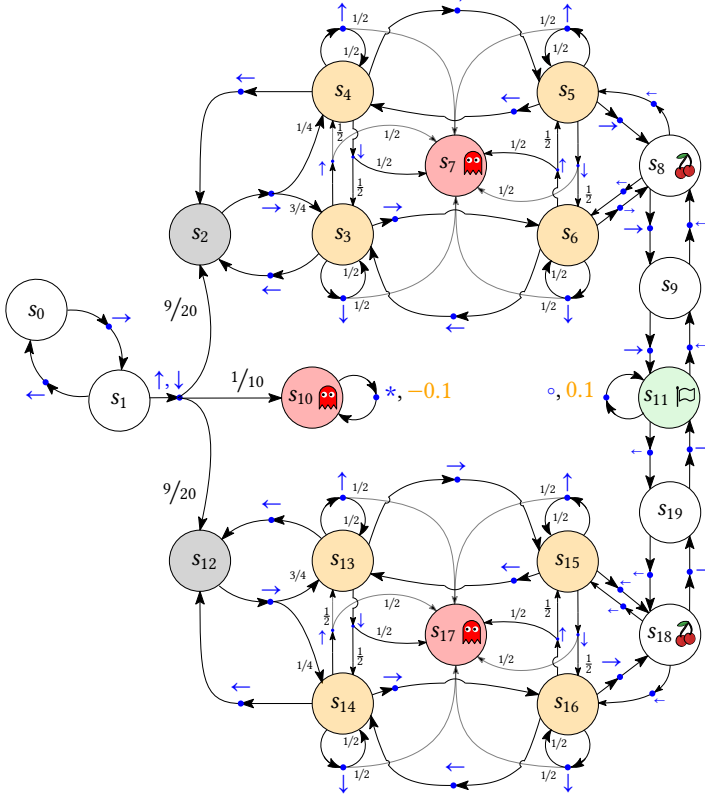
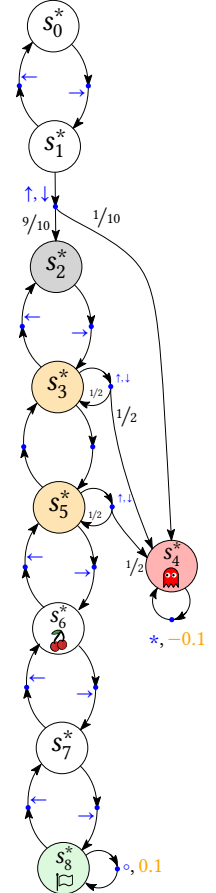
(b) MDP $\mathcal{M}_1$ representing the environment of Figure 2.13a(c) Another MDP $\mathcal{M}_2$ for the environment of Figure 2.13a such that $\mathcal{M}_1 \sim \mathcal{M}_2$

Figure 2.13: A more complex version of the grid world of Example 3 with 3 adversaries.

Consider now the relatively smaller MDP $\mathcal{M}_2$ of Figure 2.13c (only 9 states compared to the 20 states of $\mathcal{M}_1$). Interestingly, one may notice that this MDP also represents the environment of Figure 2.13a: one can easily verify that $\mathcal{M}_1$ and $\mathcal{M}_2$ are in bisimulation: $s_0^* \sim s_0$; $s_1^* \sim s_1$; $s_2^* \sim s_2 \sim s_{12}$; $s_3^* \sim s_3 \sim s_4 \sim s_{13} \sim s_{14}$; $s_4^* \sim s_7 \sim s_{10} \sim s_{17}$; $s_5^* \sim s_5 \sim s_6 \sim s_{15} \sim s_{16}$; $s_6^* \sim s_8 \sim s_{18}$; $s_7^* \sim s_9 \sim s_{19}$; and $s_8^* \sim s_{11}$.

Bisimulation implies trace and optimal return equivalence. To see why, notice that given a base B formed by equivalent classes $T_0, T_1, \dots, T_n \in \mathcal{S}/\sim$ and some measurable actions $A_0, \dots, A_{n-1} \in \Sigma(\mathcal{A})$ such that $B = T_0 \times A_0 \times \dots \times A_{n-1} \times T_n$, the countable unions, intersections, and complementations of cylinders of the form $\text{Cyl}(B)$ generate another σ -algebra over the trajectories of $\mathcal{M}$, which is included in the standard σ -algebra of $\mathcal{M}$. We call the events of this sub- σ -algebra *bisimulation-closed* events. Let π be a policy so that $\sim$ is also a bisimulation relation for $\mathcal{M}_\pi$. Then, for such any bisimulation-closed events $\mathcal{T} \subseteq \text{Traj}_{\mathcal{M}}$, we have for all states $s_1, s_2 \in \mathcal{S}$ that

$$s_1 \sim s_2 \text{ implies } \mathbb{P}_\pi^{\mathcal{M}}(\mathcal{T} \mid s_I = s_1) = \mathbb{P}_\pi^{\mathcal{M}}(\mathcal{T} \mid s_I = s_2). \quad (\text{Baier and Katoen, 2008})$$

Since the basic events (formed from bases B) of this sub- σ algebra share the same rewards and labels, bisimulation close events carry the necessary information to optimize both the expected return obtained from states in bisimulation, as well as the probability of satisfying LTL formulae.

Theorem 16: Logical and optimal return equivalence

Let $s_1, s_2 \in \mathcal{S}$ be two states in bisimulation, i.e., $s_1 \sim s_2$, then

- (i) [Larsen and Skou, 1991; Baier and Katoen, 2008]: the optimal probability (for $\text{opt} \in \{\inf, \sup\}$) of satisfying any LTL formula φ from s_1 and s_2 is the same, i.e.,

$$\text{opt}_\pi \mathbb{P}_\pi(\{\tau \mid \tau \models \varphi\} \mid s_I = s_1) = \text{opt}_\pi \mathbb{P}_\pi(\{\tau \mid \tau \models \varphi\} \mid s_I = s_2);$$

- (ii) [de Alfaro et al., 2003]: the value of any VLC formula φ are the same for s_1 and s_2 , i.e., $V(s_1, \varphi) = V(s_2, \varphi)$; and
- (iii) [Givan et al., 2003]: the optimal expected return is the same from s_1 and s_2 in $\mathcal{M}$, i.e., $V^*(s_1) = V^*(s_2)$.

Note that given any policy π with $s_1 \sim_\pi s_2$, we rather consider the bisimulation relation in $\mathcal{M}_\pi$. Then, the results enumerated in Theorem 16 are the same for $\sim_\pi$, except that the optimization term opt vanishes for LTL formulae, and the value functions considered are those of $\mathcal{M}_\pi$, i.e., V^π .

Given Theorem 16, the bisimulation relation can be leveraged to reduce the size of the MDP since the relation preserves the values functions.

Lemma 9: Model reduction

Let $\mathcal{M} = \langle S, \mathcal{A}, \mathbf{P}, \mathcal{R}, \mathbf{AP}, \ell, s_I, \gamma \rangle$ be an MDP. Consider the MDP quotient

$$\mathcal{M}/\sim = \langle S/\sim, \mathcal{A}, \mathbf{P}/\sim, \mathcal{R}/\sim, \mathbf{AP}, \ell/\sim, s_I, \gamma \rangle$$

where $(\mathbf{P}/\sim)(\cdot \mid T, a) = \mathbf{P}(\cdot \mid s, a)$, $(\mathcal{R}/\sim)(T, a) = \mathcal{R}(s, a)$, $(\ell/\sim)(T) = \ell(s)$ for any $T \in S/\sim$, $s \in T$, and $a \in \mathcal{A}$. Finally, S_I is the equivalence class containing $s_I \in S_I$. Then, by definition, $\mathcal{M}$ and $\mathcal{M}/\sim$ are equivalent under the bisimulation relation $\sim$.

The probability space of the MDP quotient $\mathcal{M}/\sim$ is the aforementioned σ -algebra generating bisimulation-closed events. Importantly, $\mathcal{M}/\sim$ preserves the optimal probability of events formalized through LTL, VLC's value functions, and the optimal expected return.

Example 16: Bisimulation quotient MDP

The MDP $\mathcal{M}_2$ of Figure 2.13c is the bisimulation quotient of the MDP $\mathcal{M}_1$ from Figure 2.13b: $s_0^* = \sim(\{s_0\})$, $s_1^* = \sim(\{s_1\})$, $s_2^* = \sim(\{s_2, s_{12}\})$, $s_3^* = \sim(\{s_3, s_4, s_{13}, s_{14}\})$, $s_4^* = \sim(\{s_7, s_{10}, s_{17}\})$, $s_5^* = \sim(\{s_5, s_6, s_{15}, s_{16}\})$, $s_6^* = \sim(\{s_8, s_{18}\})$, $s_7^* = \sim(\{s_9, s_{19}\})$, and $s_8^* = \sim(\{s_{11}\})$.

Limitations. While bisimulation offers a suitable framework to deal with large MDPs — using a smaller bisimilar model in which one can perform tractable computations — the relation is sometimes too strong to be useful. Indeed, bisimulation relations are not robust to small perturbations in the reward function or transition probabilities. To illustrate this, take the MDP of Figure 2.13c and slightly perturb the probability of going to s_4^* from s_5^* when $\uparrow$ is executed: $\mathbf{P}(s_4^* \mid s_5^*, \uparrow) = 1/2 + \varepsilon$ and $\mathbf{P}(s_5^* \mid s_5^*, \uparrow) = 1/2 - \varepsilon$ for some arbitrary small error $\varepsilon > 0$. While $\mathcal{M}_1$ and $\mathcal{M}_2$ remain close to each other at first glance after applying this perturbation, they are not bisimilar anymore and are treated, in terms of bisimulation, as being completely different. This is very problematic, especially if one hopes, e.g., to learn a bisimilar MDP from data, where it is not possible in general to learn perfect distributions without any error.

This motivates us to further investigate another *non-binary* metric, which would intuitively assign a softer, real-valued “score” of bisimilarity between states instead of strictly classifying them as being bisimilar or not. To define this metric, we need to identify the perturbations which cause the original relation to fail and quantify them. Quantifying a discrepancy between real values (as the rewards) is trivial — one simply

needs to consider the Euclidean distance — but this is not so evident with probability distributions.

4.2 Probability Metrics

As we already discussed, “quantifying the bisimilarity” between states of one or multiple MDPs necessarily involves quantifying the discrepancy of their transition distributions. In the following, we introduce several discrepancies between probability measures and discuss how they are related, their benefits, and their drawbacks in the context of defining a bisimulation metric for MDPs. We start with the notion of *metric space*.

Definition 31: Distance and metric spaces

Let $\mathcal{X}$ be a measurable set and $d : \mathcal{X} \rightarrow [0, \infty)$. Then, we say that $\langle \mathcal{X}, d \rangle$ is a *metric space* if and only if d is a *metric* on $\mathcal{X}$. Precisely, d must fulfill the following atoms for all $x, y, z \in \mathcal{X}$.

1. *Identity of indiscernibles*: $d(x, y) = 0$ if and only if $x = y$.
2. *Symmetry*: $d(x, y) = d(y, x)$.
3. *Triangle inequality*: $d(x, z) \leq d(x, y) + d(y, z)$.

Along this section, we fix the measurable set $\mathcal{X}$ and we assume it is equipped with a *distance metric* d , leading to the metric space $\langle \mathcal{X}, d \rangle$.

One of the most popular discrepancies used in information theory to compare probability measures and data distributions is the *Kullback-Leibler divergence*.

Definition 32: Kullback-Leibler divergence

Let $P, Q \in \Delta(\mathcal{X})$ be two probability measures over $\mathcal{X}$, then the *Kullback-Leibler divergence* (KL) is given by

$$D_{\text{KL}}(P \parallel Q) = \int_{\mathcal{X}} \log \left(\frac{dP}{dQ} \right) dP(x),$$

where dP/dQ denotes the standard Radon-Nikodym derivative of P w.r.t. Q . Notably, when (i) $\mathcal{X}$ is countable, $D_{\text{KL}}(P \parallel Q) = \mathbb{E}_{x \sim P} [\log(P(x)/Q(x))]$; and (ii) p, q are respectively density functions of P and Q , $D_{\text{KL}}(P \parallel Q) = \mathbb{E}_{x \sim P} [\log(p(x)/q(x))]$.

KL stands out for its simplicity and its ease of optimization. However, although always non-negative, *KL is not a metric* on $\Delta(\mathcal{X})$. Indeed, the symmetry (2) and triangle inequality (3) are not fulfilled. Importantly, KL may be undefined when Q does not have the same

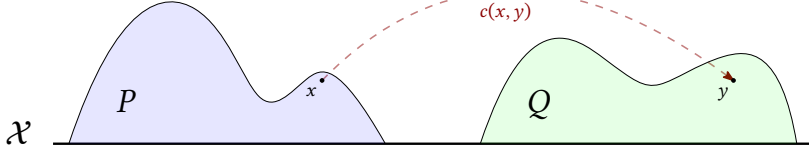


Figure 2.14: Optimal transport problem from P to Q with cost function c .

support as P : instabilities arise in the quotient P/Q when $Q(E) = 0$ for events $E \in \Sigma(\mathcal{X})$ drawn under P . Given these limitations, KL is ill-suited in the context of comparing transition probabilities to quantify bisimilarity. Nonetheless, as we will see later in this section and the subsequent chapters of this thesis, KL remains a compelling discrepancy measure with theoretical connections to probability metrics.

Optimal transport. In the following, we discuss a family of probability metrics related to the *optimal transport problem* (OT) [Villani, 2009]. The optimal transport, originally formalized by Monge [1781], is a classical problem in mathematics and economics, which aims to find the most efficient way to transform a distribution P to another distribution Q , where the cost of moving a particle from a position x in the support of P to a position y in the support of Q is given by a cost function $c(x, y)$.

Definition 33: Optimal transport problem

Let $P, Q \in \Delta(\mathcal{X})$ be two probability measures over $\mathcal{X}$. The *optimal transport problem* (OT) from P to Q with *cost function* $c : \mathcal{X} \times \mathcal{X} \rightarrow [0, \infty)$ is defined as

$$\mathcal{W}_c(P, Q) = \inf_{\omega \in \Pi(P, Q)} \int_{\mathcal{X} \times \mathcal{X}} c(x, y) \cdot \omega(dx, dy) = \inf_{\omega \in \Pi(P, Q)} \mathbb{E}_{x, y \sim \omega} c(x, y),$$

where

$$\Pi(P, Q) = \left\{ \omega \in \Delta(\mathcal{X} \times \mathcal{X}) \mid \forall E \in \Sigma(\mathcal{X}), P(E) = \int_{\mathcal{X}} \omega(E, dy) \text{ and } Q(E) = \int_{\mathcal{X}} \omega(dx, E) \right\}$$

is the set of all *couplings* (also called *transport plans*) between P and Q (the set of measures over $\mathcal{X} \times \mathcal{X}$ with marginals P and Q).

At a very intuitive level, if one sees the two distributions P, Q as piles of dirt (see Figure 2.14), the OT captures the minimum amount of work to transform the pile of dirt P into the pile of dirt Q . When the cost function c coincides with the distance metric d associated with the measurable space $\mathcal{X}$, the OT defines a *distance metric* on $\Delta(\mathcal{X})$, called

the *Wasserstein distance* [Vaserstein, 1969]. In that case, the Wasserstein distance can be re-formalized via the Kantorovich and Rubinstein [1958] duality:

Theorem 17: Kantorovich and Rubinstein [1958] duality

$$\mathcal{W}_d(P, Q) = \sup_{f \in \text{Lipsch}_d^1} \mathbb{E}_{x \sim P} f(x) - \mathbb{E}_{y \sim Q} f(y)$$

where Lipsch_d^1 denotes the set of all 1-Lipschitz functions w.r.t. the metric d :

$$\text{Lipsch}_d^1 = \{f : \mathcal{X} \rightarrow \mathbb{R} \mid f(x) - f(y) \leq d(x, y) \text{ for all } x, y \in \mathcal{X}\}.$$

Remark 14: Dual formulation of Wasserstein and absolute value

In the literature, the definition may sometimes slightly differ, making use of an absolute value in the dual formulation of the Wasserstein distance, i.e.,

$$\mathcal{W}_d(P, Q) = \sup_{f \in \text{Lipsch}_d^1} \left| \mathbb{E}_{x \sim P} f(x) - \mathbb{E}_{y \sim Q} f(y) \right|.$$

This is equivalent to the formulation which does not make use of the absolute value. To see why, assume on the contrary that $\sup_f |\mathbb{E}_P[f] - \mathbb{E}_Q[f]| \neq \sup_f \mathbb{E}_P[f] - \mathbb{E}_Q[f]$. Let us consider the case where $\sup_f |\mathbb{E}_P[f] - \mathbb{E}_Q[f]| < \sup_f \mathbb{E}_P[f] - \mathbb{E}_Q[f]$. This is clearly not possible: just take f^* as the function that attains the supremum of the right hand side of the inequality, then

$$\sup_{f \in \text{Lipsch}_d^1} \left| \mathbb{E}_{x \sim P} f(x) - \mathbb{E}_{y \sim Q} f(y) \right| < \left| \mathbb{E}_{x \sim P} f^*(x) - \mathbb{E}_{y \sim Q} f^*(y) \right| = \mathbb{E}_{x \sim P} f^*(x) - \mathbb{E}_{y \sim Q} f^*(y),$$

which is a contradiction. For the other inequality, $\sup_f |\mathbb{E}_P[f] - \mathbb{E}_Q[f]| > \sup_f \mathbb{E}_P[f] - \mathbb{E}_Q[f]$, take f^* to be the function which attains the supremum the left-hand side of the inequality. On the one hand, when $\mathbb{E}_P[f^*] \geq \mathbb{E}_Q[f^*]$, then we have $|\mathbb{E}_P[f^*] - \mathbb{E}_Q[f^*]| = \mathbb{E}_P[f^*] - \mathbb{E}_Q[f^*] > \sup_f \mathbb{E}_P[f] - \mathbb{E}_Q[f]$, which is already a contradiction. On the other hand, when $\mathbb{E}_P[f^*] < \mathbb{E}_Q[f^*]$, we have $|\mathbb{E}_P[f^*] - \mathbb{E}_Q[f^*]| = \mathbb{E}_Q[f^*] - \mathbb{E}_P[f^*]$. Take $f' = -f^*$, f' is also 1-Lipschitz by definition since $f^*(x) - f^*(y) \leq d(x, y)$ and $f^*(y) - f^*(x) \leq d(x, y)$. Then, we have $|\mathbb{E}_P[f^*] - \mathbb{E}_Q[f^*]| = \mathbb{E}_P[f'] - \mathbb{E}_Q[f'] > \sup_f \mathbb{E}_P[f] - \mathbb{E}_Q[f]$, leading to another contradiction.

Thus, the definitions with and without the absolute value are equivalent. Along this thesis, we may use them interchangeably.

Remark 15: Various names for the same metric

The Wasserstein distance appears under a variety of names in the literature, including the *Kantorovich*, *Kantorovich-Rubinstein*, *Monge-Kantorovich*, *Hutchinson*, or *earth mover's* distance. In this dissertation, we use *Wasserstein* in reference to the Russian mathematician Leonid Vaserstein (Wasserstein is the German spelling), who introduced this distance in the context of Markov processes and automata games [Vaserstein, 1969].

When the metric space $\langle \mathcal{X}, d \rangle$ is discrete, the distance d is generally assumed to be the *discrete metric* $\mathbb{1}_{\neq}$, defined as $\mathbb{1}_{\neq}(x, y) = \mathbb{1}\{x \neq y\}$. Remarkably, in that case, the Wasserstein distance $\mathcal{W}_{\mathbb{1}_{\neq}}(P, Q)$ coincides with the *total variation distance* [Villani, 2009].

Definition 34: Total variation

Given two probability measures $P, Q \in \Delta(\mathcal{X})$, the *total variation distance* (TV) between P and Q is defined as

$$d_{TV}(P, Q) = \sup_{A \in \Sigma(\mathcal{X})} |P(A) - Q(A)| = \mathcal{W}_{\mathbb{1}_{\neq}}(P, Q).$$

Notably, when (i) $\mathcal{X}$ is countable, $d_{TV}(P, Q) = \frac{1}{2} \sum_{x \in \mathcal{X}} |P(x) - Q(x)|$; and (ii) p, q are respectively density functions for P and Q , $d_{TV}(P, Q) = \frac{1}{2} \int_{\mathcal{X}} |p(x) - q(x)| dx$.

One can also see d_{TV} as an *integral probability metric*, where

$$d_{TV}(P, Q) = \frac{1}{2} \sup_{f \in \mathcal{F}} \left| \mathbb{E}_{x \sim P} f(x) - \mathbb{E}_{y \sim Q} f(y) \right|,$$

where $\mathcal{F} = \{f : \sup_{x \in \mathcal{X}} |f(x)| \leq 1\}$ is the set of one-bounded functions [Müller, 1997; Sriperumbudur et al., 2009].

TV is also related to the KL divergence by the Pinsker's inequality [Pinsker, 1964].

Theorem 18: Pinsker's inequality

Given $P, Q \in \Delta(\mathcal{X})$, the total variation distance is bounded by KL as follows:

$$d_{TV}(P, Q) \leq \sqrt{\frac{1}{2} D_{KL}(P \parallel Q)}.$$

Hence, when considering the Wasserstein distance in the discrete setting, one can always derive an upper bound based on the KL divergence, leveraging the Pinsker's inequality.

To compare probability transition functions induced by different states, we show in the following that the Wasserstein distance is well-suited and offers a fixed point interpretation of bisimulation metrics.

4.3 Bisimulation Pseudometrics

Our objective now is to define a measure for quantifying the distance between states and models, which captures the essence of the bisimulation relation. However, the conventional metric concept is not ideal for this due to the identity of indiscernible requirement (Definition 31.1), which may not hold in our context — a zero distance should characterize the bisimulation relation instead of the equality relation. Thus, we turn to the notion of a *pseudometric*, which relaxes this requirement and allows for a more flexible characterization of similarity.

Definition 35: Pseudometrics

Let $d : \mathcal{X} \times \mathcal{X} \rightarrow [0, \infty)$ be a function on $\mathcal{X}$. Then, d is a *pseudometric* on $\mathcal{X}$ if and only if it satisfies the symmetry and triangle inequality atoms of Definition 31.2 and 31.3. Moreover, the identity of indiscernibles (31.1) is relaxed to the following atom:

$$x = y \text{ implies } d(x, y) = 0 \text{ for all } x, y \in \mathcal{X}.$$

Fixed point theory. To define a pseudometric capturing the bisimulation relation, it is useful to see the bisimulation relation in terms of *fixed point*. For this purpose, we need to carefully handle the case where the state space is continuous and make the following additional assumptions.

Assumption 1: Bisimulation and continuous state space

When the state space is a Borel set, we additionally require the following assumptions. For all $a \in \mathcal{A}$,

- (i) $\mathcal{R}(\cdot, a)$ is continuous on S ;
- (ii) $P(\cdot \mid s, a)$ is weakly continuous w.r.t. $s \in S$, i.e., for all sequences $(s_n)_{n \in \mathbb{N}}$ converging to s , $P(\cdot \mid s_n, a)$ weakly converges to $P(\cdot \mid s, a)$ as $n \rightarrow \infty$.

Assumption (ii) is a typical mild requirement imposed in scenarios involving reasonable systems. Assumption (i) is less common but still relatively mild. Although it may initially appear restrictive, in practice, it is quite manageable. Generally, the reward function is not assumed to be continuous across the entire state space. However, it is often

assumed to have only a finite number of discontinuities. A notable example lies in goal-oriented tasks, where the agent receives a sparse reward upon reaching a goal state. In such scenarios, it is sufficient to smooth the reward function by applying a sigmoid function ($\sigma(x) = 1/(1+\exp(-x))$) at the discontinuity points. For further discussions about the requirements of these assumptions, we refer the reader to the work of Ferns et al. [2005].

Theorem 19: Fixed point interpretation of bisimulation

Define the *bisimulation operator* $\mathcal{F} : 2^{S \times S} \rightarrow 2^{S \times S}$ as

$$s_1 \mathcal{F}(\mathcal{R}) s_2$$

if and only if

$$\forall a \in \mathcal{A}, \forall T \in \mathcal{S}/\mathcal{R}, \ell(s_1) = \ell(s_2), \mathcal{R}(s_1, a) = \mathcal{R}(s_2, a), \text{ and } \mathbf{P}(T \mid s_1, a) = \mathbf{P}(T \mid s_2, a)$$

for all relation $\mathcal{R}$ on S and $s_1, s_2 \in S$. Then $\sim$ is the greatest fixed point of $\mathcal{F}$.

This theorem is a direct consequence of the fact that $\mathcal{F}$ is a *monotone* mapping: when starting from a coarse relation, each application of $\mathcal{F}$ necessarily refines the relation. Then, by the Knaster-Tarski theorem [Tarski, 1955], the greatest fixed point of $\mathcal{F}$ is guaranteed to exist and coincides with the bisimulation relation by definition. For instance, starting from the relation $s_1 \mathcal{R}_0 s_2 \iff \ell(s_1) = \ell(s_2)$ and refining it through $\mathcal{R}_{n+1} = \mathcal{F}(\mathcal{R}_n)$ guarantees the convergence to the bisimulation relation $\sim$ as $n \rightarrow \infty$.

To extend binary relations to distances, we similarly define a fixed point operator on pseudometrics by replacing each equality in the definition by distance metrics: the discrete metric for the labels, the usual Euclidean distance for the rewards, and the Wasserstein distance for the probability transition functions. To derive a set of significant properties, we additionally differentiate between pseudometrics based on labels and rewards, as outlined in Remark 13. We further use the symbol $\clubsuit \in \{\ell, \mathcal{R}\}$ to denote the type of bisimulation pseudometric that we are considering — ℓ for the pseudometrics involving labels and $\mathcal{R}$ for those involving rewards.

Theorem 20: Bisimulation pseudometrics

Let $\clubsuit \in \{\ell, \mathcal{R}\}$, $B = \sup_{s, s', a} |\mathcal{R}(s, a) - \mathcal{R}(s', a)|$, and $p\mathcal{M}_S$ denote the space of *bounded* pseudometrics on S (one-bounded when $\clubsuit = \ell$, and by $B/(1-\gamma)$ otherwise). Define the bisimulation operator $\mathcal{F}^\clubsuit : p\mathcal{M}_S \rightarrow p\mathcal{M}_S$ as

$$(\mathcal{F}^\clubsuit d)(s_1, s_2) = \begin{cases} \max \left(\mathbb{1}_{\neq}(\ell(s_1), \ell(s_2)), \sup_{a \in \mathcal{A}} \gamma \cdot \mathcal{W}_d(\mathbf{P}(\cdot \mid s_1, a), \mathbf{P}(\cdot \mid s_2, a)) \right) & \text{if } \clubsuit = \ell, \text{ and} \\ \sup_{a \in \mathcal{A}} |\mathcal{R}(s_1, a) - \mathcal{R}(s_2, a)| + \gamma \cdot \mathcal{W}_d(\mathbf{P}(\cdot \mid s_1, a), \mathbf{P}(\cdot \mid s_2, a)) & \text{otherwise, when } \clubsuit = \mathcal{R}. \end{cases}$$

for all $d \in p\mathcal{M}_S$ and $s_1, s_2 \in S$. Then, $\tilde{d}^\clubsuit$ is the unique fixed point of $\mathcal{F}^\clubsuit$.

This time, due to the introduction of discounting, $\mathcal{F}^\clubsuit$ is a monotone *contraction* mapping. Thus, a unique fixed point $\tilde{d}^\clubsuit$ is guaranteed by the Banach's fixed point theorem.

We call $\tilde{d}$ the *bisimulation pseudometric*, or more soberly the *bisimulation distance*. As usual, we denote by $\tilde{d}_\pi$ the bisimulation distance defined in the MC $\mathcal{M}_\pi$ induced by a given policy π . As in the case of bisimulation relations, the distance can be extended to compare MDPs by considering the disjoint union of the state spaces of the two models (see Definition 30).

Remarkably, the *kernel* of the bisimulation pseudometrics — where the distance equals zero — characterizes the bisimulation relation. Crucially, the value functions presented throughout this thesis are Lipschitz continuous with respect to the bisimulation distance.

Theorem 21: Properties of bisimulation pseudometrics

Denote by $\sim^\ell$ and $\sim^\mathcal{R}$ the largest bisimulation relations obtained by respectively relaxing the requirement involving the reward and the labels equality (cf. Remark 13). Then, let $s_1, s_2 \in S$ be two states of $\mathcal{M}$ and $\clubsuit \in \{\ell, \mathcal{R}\}$ we have

- (i) $\tilde{d}^\ell$ corresponds to the *largest behavioral difference* [de Alfaro et al., 2003; Desharnais et al., 2004]:

$$\tilde{d}^\ell(s_1, s_2) = \sup_{\varphi \in \text{VLC}(\gamma)} |V(s_1, \varphi) - V(s_2, \varphi)|,$$

where $\text{VLC}(\gamma)$ is the set of all closed VLC formulae using discount factors lower or equal to γ . Then, *the value of every formula is Lipschitz continuous w.r.t. $\tilde{d}^\ell$* ;

Theorem 21: Properties of bisimulation pseudometrics (*continued*)

(ii) *the optimal expected return is Lipschitz continuous w.r.t. $\tilde{d}^R$* [Ferns et al., 2005]:

$$|V^*(s_1) - V^*(s_2)| \leq \tilde{d}^R(s_1, s_2); \text{ and}$$

(iii) *the kernel of $\tilde{d}^\star$ is bisimilarity* [de Alfaro et al., 2003; Desharnais et al., 2004; Ferns et al., 2005]:

$$\tilde{d}^\star(s_1, s_2) = 0 \text{ if and only if } s_1 \sim^\star s_2.$$

We stress that the actual numerical value of the bisimulation distance is not the most important. Instead, the key point is that the bisimulation distance is a pseudometric capturing the essence of the largest bisimulation relation and that the value functions are Lipschitz continuous with respect to it. The most crucial aspects of the distance lie in its properties: the kernel of the distance corresponds to the bisimulation relation, states that are close in distance exhibit similar values, and the metric demonstrates robustness against perturbations.

5 Concluding Discussion

The bisimulation pseudometric is a robust tool for comparing states and MDPs, extending the concept of the bisimulation relation.

In the context of RL, it gains significance when learning *state space representations* (i.e., functions of the form $\phi : \mathcal{S} \rightarrow \bar{\mathcal{S}}$, where $\bar{\mathcal{S}}$ “represents” $\mathcal{S}$). If a representation captures the bisimulation pseudometric — implying that close points in the representation space would correspond to a close bisimulation distance — it becomes particularly interesting for learning a policy (thus conditioning the latter on $\bar{\mathcal{S}}$ rather than on $\mathcal{S}$). Indeed, such a representation would definitely facilitate the learning of the policy since its underlying metric would capture optimal value functions through its Lipschitz continuity properties.

In the context of verification, the pseudometric allows deriving guarantees on the interaction of the agent with the environment by model checking a tractable, bisimilarly close model that is possibly prone to perturbations. Models learned from agent-environment interactions are indeed prone to perturbations, making a closely bisimilar model well suited for reliably verifying the behaviors of the agent in RL scenarios.

These observations set the context for this thesis: our objective is to develop algorithms for learning tractable representations of the state space that effectively capture the bisimulation distance, aligning with the principles of model checking. Additionally, we aim to provide guarantees regarding the reliability of the learned representation despite

the possible perturbations induced by the learning process. In subsequent chapters, we delve into addressing these challenges and conduct an extensive analysis of the learned representations. Specifically, we endeavor to shed light on the following questions:

1. *How can we effectively learn a representation of the state space that accurately captures the bisimulation pseudometric in the context of RL?*
2. *How to provide guarantees on the learned representation and assess its robustness to perturbations?*
3. *How can we harness the learned representation to verify the behaviors of an agent operating under a learned RL policy?*
4. *Is it feasible to leverage this representation to facilitate the efficient learning of a policy?*

By rigorously exploring those questions, we hope to contribute to advancing the understanding and application of learned representations in the context of formal verification and RL.

Distillation of RL Policies through Verifiable Latent Space Models

In the preceding chapter of this thesis, we introduced several concepts that are not typically united. We have seen that *reinforcement learning* (RL) can be used to train agent’s policies for complex tasks, in scenarios where an explicit model of the environment is not necessarily provided, but wherein the agent is capable of interacting, exploring, and observing the outcome of its decisions. State-of-the-art RL algorithms leverage *deep neural networks* to scale to tasks involving general state spaces — “deep” stands for the ability of those neural networks to learn and discover a useful and tractable *representation* of input data. Those algorithms are grouped together under the term *deep RL* (DRL). DRL has garnered attention for its exceptional performance and applicability to real-world scenarios [Mason and Grijalva, 2019; Bellemare et al., 2020; Libin et al., 2020; Ceusters et al., 2021; Reymond et al., 2024; Degraeve et al., 2022; Wurman et al., 2022]. However, we highlighted a caveat: the introduction of deep neural networks compromises the reliability of those algorithms along with their performance and convergence guarantees (cf. Chapter 2, Section 2.4 for a detailed discussion).

On the other hand, formal methods — and more precisely *model checking* principles — offer a different approach when a tractable environmental model is available. These methods enable the derivation of rigorous guarantees for agents operating under (finite-memory) policies. While learning *undiscounted* linear properties by interacting with the environment is not possible, discounted or finite horizon properties are in contrast learnable from data. Moreover, with access to the model of the environment, approximate values for discounted properties can be computed using geometrically

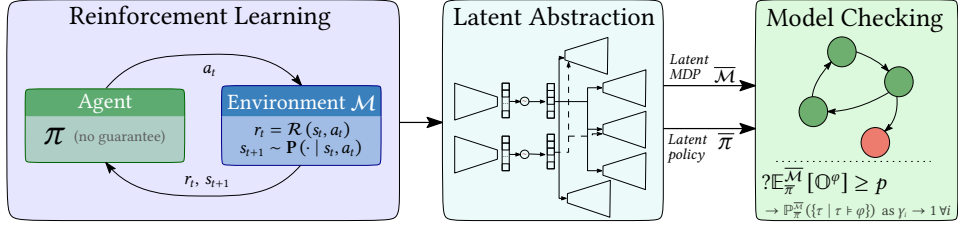


Figure 3.1: Big picture of the distillation approach. First, a proposal policy π is learned using a DRL algorithm in the original environment $\mathcal{M}$, without any guarantee on the agent’s resulting behavior. Next, a latent MDP $\bar{\mathcal{M}}$ is learned based on the execution of π within $\mathcal{M}$, and the policy is distilled over the resulting latent spaces, producing a new latent policy $\bar{\pi}$. Since the latent spaces are now finite, model checking is applied to verify $\bar{\pi}$ in $\bar{\mathcal{M}}$ for given properties φ of interest. The guarantees obtained for the latent policy can then be lifted to the original environment $\mathcal{M}$ via the established bisimulation bounds between $\mathcal{M}_{\bar{\pi}}$ and $\bar{\mathcal{M}}_{\bar{\pi}}$ — notice here that $\bar{\pi}$ is designed to be compliant with $\mathcal{M}$.

converging algorithms like value iteration. Notably, the set of discounted properties characterizes *bisimulation pseudometrics*, quantifying the behavioral differences between systems based on the properties they satisfy. Namely, the bisimulation distance is defined as the supremum of the value difference of the specification they satisfy, ranging over all discounted properties — the distance is one if the models satisfy strictly different properties, and zero if they satisfy exactly the same properties (cf. Theorem 21).

This chapter introduces methods involving the combination of RL and model checking, leveraging the scalability and performance of DRL alongside the reliability and guarantees provided through formal methods. Precisely, we tackle the problem of providing policies learned via (any) DRL algorithm with formal guarantees, for agents operating in environments involving general state-action spaces. To do so, our approach consists of model checking a *bisimilarly close latent space model* representing the interaction of the agent and the environment. Notably, we learn an abstract, simpler, and tractable model of the original environment through *state and action embedding functions*, mapping each state and action of the original environment to a *latent space*. We term the resulting model a *latent MDP*. Since the execution of the policy learned through the DRL algorithm is not possible in the latent model due to the introduction of the new abstract spaces, we *distill the original RL policy* over the latent spaces, resulting in a new *latent policy*, which is, in contrast, executable in the original and latent models *and* verifiable. The high-level idea behind this approach is sketched in Figure 3.1.

We enable the verification of the latent policy thanks to the following ideas:

-
- (i) we learn a *discrete*, finite latent space model of the original environment, making it amenable to model checking;
 - (ii) we provide bisimulation guarantees between the original and latent models through *local losses* and probably approximately correctly (PAC) bounds that quantify the quality of the resulting abstraction and representation;
 - (iii) the loss functions¹ used to learn the latent spaces embed upper bounds on the bisimulation pseudometrics, either indirectly (through variational auto-encoders) or directly — by optimizing a penalized version of the optimal transport problem (OT).

We provide a comprehensive analysis of the proposed approach, including theoretical guarantees, the introduction of two generative models (respectively based on variational lower bounds and the OT), as well as experimental results. We first dwell on the theoretical aspects of the approach and present *abstraction and representation quality guarantees* on the learned model. Those guarantees are formalized through PAC bounds on bisimulation distances that quantify the behavioral difference between the original and latent space models. Second, we introduce principled generative models that learn an abstract model of the original environment with bisimulation guarantees.

Origins of the results. The results presented in this chapter come from those established in the articles [Delgrange et al., 2022, 2023a, 2024]. Their presentation is adapted to the context of this thesis, and the content is extended with additional explanations and discussions. Unpublished theoretical contributions are marked by the symbol $\star$.

Related Work

Formal methods in RL. Several complementary approaches address safe RL through formal methods [Junges et al., 2016; Alshiekh et al., 2018; Jansen et al., 2020; Simão et al., 2021; Chakraborty et al., 2023], with the aim of ensuring safety *during* RL. These works typically assume prior access to an abstract model capturing the safety aspects of the environment. Additionally, Alamdari et al. [2020] apply synthesis and model checking to policies distilled from deep RL, albeit without abstraction or representation quality guarantees. Some frameworks pursue the verification of deep RL policies [Bacci and Parker, 2020, 2022; Carr et al., 2020], but rely on known environment models and other assumptions, such as deterministic or discrete environments. Finally, *DeepSynth* [Hasanbeig et al., 2021] enables learning a formal model from execution traces, with the distinct aim of guiding agents toward sparse and non-Markovian rewards.

¹the objective functions established to optimize the function approximators that yield the latent space.

Representation learning. Works on *representation learning* in the context of RL [Gelada et al., 2019; van der Pol et al., 2020; Castro et al., 2021; Zhang et al., 2021; Zang et al., 2022] consider bisimulation metrics to optimize the representation quality, and aim at learning (continuous) representations that capture bisimulation, so that two states close in the representation are guaranteed to provide close and relevant information to optimize the performance of the controller. In particular, *DeepMDPs* [Gelada et al., 2019] are learned by optimizing similar losses to ours, but do not provide verifiable confidence measurement and have only been evaluated on deterministic MDPs (we elaborate on the difference of our work with DeepMDPs in Remark 17).

Model learning. Other model-based approaches learn a latent representation of the environment and train (possibly simpler) policies from the extracted features [Corneil et al., 2018; Ha and Schmidhuber, 2018; François-Lavet et al., 2019; Freeman et al., 2019; Lee et al., 2020]. Notably, Corneil et al. [2018] and Hafner et al. [2021] adopt a similar approach to ours of learning discrete latent models from continuous-state environments to plan in the latent space, but do not offer formal guarantees or involve policy distillation. Furthermore, Hafner et al. [2021] suggest that learning discrete latent variables may be beneficial for generalization and optimization, particularly in scenarios with “non-smooth” transitions (i.e., where the distributions exhibit discontinuities).

In the context of agent-based simulation, causal models can be used as an alternative to MDPs to define different layers of abstraction of the interaction between the agent and environment, aiding reasoning on computationally intensive simulators. Notably, Zennaro et al. [2023]; Dyer et al. [2023] explore various discrepancy metrics (e.g., KL, Jensen-Shannon) to learn abstract models and compare their probability measures to those of the original environment, thereby deriving properties to establish bounds on model discrepancy.

1 Latent Space Modeling

In the context of this thesis, we define a *latent space model* as an explicit model that aims to *represent* the environment, the latter being possibly unknown, large, or even defined over continuous state or action spaces. The latent model provides a simplified, abstract, and tractable representation of the original environment. In RL, this model is typically learned from data collected during the agent’s interaction with the environment. Importantly, explicit computations are made possible in latent MDPs through access to their latent reward and transition functions. The accuracy of the latent representation as a model of the original environment hinges on how these reward and transition functions are learned. Notably, we demonstrate in this section that by selecting an appropriate

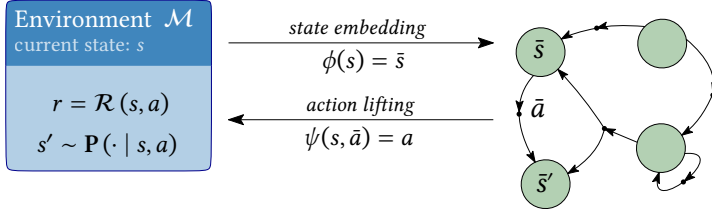


Figure 3.2: Original environment (left) and some discrete latent MDP (right). Both models are linked via state embedding and action lifting functions (respectively, ϕ and ψ).

discrepancy measure between these functions, connections can be established with bisimulation pseudometrics.

Formally, a latent space model is defined as follows.

Definition 36: Latent space model

Given the original (possibly unknown, large, or continuous) environment $\mathcal{M} = \langle S, \mathcal{A}, \mathbf{P}, \mathcal{R}, \mathbf{AP}, \ell, s_I, \gamma \rangle$, a latent space model consists of another (simpler, tractable, and explicit) latent MDP $\bar{\mathcal{M}} = \langle \bar{S}, \bar{\mathcal{A}}, \bar{\mathbf{P}}, \bar{\mathcal{R}}, \bar{\mathbf{AP}}, \bar{\ell}, \bar{s}_I, \gamma \rangle$ whose state and action spaces are linked to the original ones via a *state embedding functions* $\phi : S \rightarrow \bar{S}$ as well as an *action lifting function* $\psi : S \times \bar{\mathcal{A}} \rightarrow \mathcal{A}$. We further assume that (i) $\bar{S}$ is equipped with a distance metric $\bar{d}$, and (ii) as the original reward function, $\bar{\mathcal{R}}$ is also bounded by $R_{\max}$. We denote by $\bar{V}$ the value functions of the latent MDP $\bar{\mathcal{M}}$.

The links drawn between the original and latent spaces by the state embedding and action lifting functions are illustrated in Figure 3.2. Interestingly, the connections established between the original and latent spaces allow to *lift* a latent policy $\bar{\pi}$ defined for $\bar{\mathcal{M}}$ to the original environment $\mathcal{M}$. This lifting is illustrated in Figure 3.3: each time the agent is located in a concrete state $s \in S$ of the original environment, s is embedded into the latent space via $\phi(s) = \bar{s}$. Then, based on this latent state $\bar{s}$, the latent policy $\bar{\pi}$ prescribes the next action $\bar{a} \sim \bar{\pi}(\cdot | \bar{s})$ to be executed in the latent model. This action is then lifted back to the original environment via $\psi(s, \bar{a}) = a$. One may then consider $\bar{\pi}$ as a lifted policy for $\mathcal{M}$: by abuse of notation, we write $\bar{a} \sim \bar{\pi}(\cdot | s)$

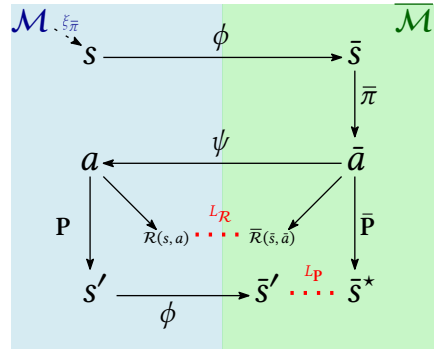


Figure 3.3: Lifted policy and local losses.

to first sample the latent action $\bar{a}$ from $\bar{\pi}(\cdot \mid \phi(s))$, and $a \sim \pi(\cdot \mid s)$ for lifting it afterward to the original action space using $a = \psi(s, \bar{a})$.

Remark 16: Discrete actions

Assume that the original action space $\mathcal{A}$ is a (tractable) finite set. Then, it is often not necessary to consider abstracting the action space, and the action lifting function ψ can be merely taken as the identity function: $\psi(\cdot, \bar{a}) = \bar{a}$ for all $\bar{a} \in \bar{\mathcal{A}}$, with $\bar{\mathcal{A}} = \mathcal{A}$.

From that point, one can then *measure* the distance between the original and latent reward and transition functions by comparing the resulting behavior of the agent in the two models.

On the one hand, a reward $\mathcal{R}(s, a)$ is obtained in the original environment $\mathcal{M}$ when the agent executes the action a in state s . In parallel, a reward $\bar{\mathcal{R}}(\bar{s}, \bar{a})$ is obtained in the latent model $\bar{\mathcal{M}}$ when the agent executes the action $\bar{a}$ in state $\bar{s}$. By abuse of notation, we write $\mathcal{R}(s, \bar{a})$ for the reward obtained in the original environment when $\bar{a}$ is lifted from the latent space to the original one, i.e., $\mathcal{R}(s, \psi(s, \bar{a}))$. Then, the distance between the two rewards is merely given by $|\mathcal{R}(s, \bar{a}) - \bar{\mathcal{R}}(\bar{s}, \bar{a})|$.

On the other hand, the two models transition to their next state $s' \sim \mathbf{P}(\cdot \mid s, a)$ and $\bar{s}' \sim \bar{\mathbf{P}}(\cdot \mid \bar{s}, \bar{a})$. To enable the comparison between $\mathbf{P}$ and $\bar{\mathbf{P}}$, one needs to embed s' into the latent space and compare the embedded state $\phi(s') = \bar{s}'$ with $\bar{s}'$. Again by abuse of notation, we write $\phi\mathbf{P}(\cdot \mid s, \bar{a})$ for the resulting pushforward measure, i.e., drawing s' from $\mathbf{P}(\cdot \mid s, \psi(s, \bar{a}))$ and then embedding s' into the latent space via $\bar{s}' = \phi(s')$. We formalize the resulting metric as the *Wasserstein distance* between the resulting pushforward measure and the latent transition function, i.e., $\mathcal{W}_{\bar{a}}(\phi\mathbf{P}(\cdot \mid s, \bar{a}), \bar{\mathbf{P}}(\cdot \mid \bar{s}, \bar{a}))$.

Remark 17: Deep MDPs

Those formulations of distance metrics between reward and transition functions have been explored by Gelada et al. [2019] in the context of representation learning for RL. They minimized these losses to learn *DeepMDPs*, which are latent models for environments with discrete action spaces. DeepMDPs rely on some Lipschitz continuity assumptions applied to the latent reward and transition functions. Notably, those Lipschitz constraints need to be enforced at training time, when learning those functions through neural networks. When the Lipschitz constant is sufficiently small, bisimulation bounds can be established according to the derived losses in the *global setting*. In that case, the bisimulation distance is considered over the entire state-action space — a computationally infeasible task in practice. Enforcing Lipschitz continuity of the networks — especially for the transition function involving computation with the Wasserstein metric — presents significant

challenges.^a To address the difficulty of estimating the Wasserstein metric in practice, Gelada et al. assumed deterministic MDPs (where $\mathbf{P}$ is Dirac).

In contrast, we consider MDPs with general action spaces and stochastic transition functions. Our forthcoming bounds do not require any Lipschitz continuity assumption on the latent functions. Additionally, we provide PAC guarantees to quantify the quality of the learned latent spaces. Lastly, as detailed in the following sections, we focus on the *local setting*.

^awe refer the interested reader to the thesis of Genevay [2019] who covered the challenges of learning (variations of the) Wasserstein measure from data.

1.1 Local Losses

We particularly focus on computing those metrics *locally*, i.e., under the execution of a given policy in the original environment $\mathcal{M}$ — in contrast to *globally*, where the distance needs to be computed over the entire state-action space. There are compelling reasons for considering local metrics. First, local metrics are amenable to computation and optimization, particularly through neural networks and gradient descent methods, as they are defined over states and actions that are actually visited by the agent. This stands in stark contrast to global metrics, which, while theoretically intriguing, are computationally intractable in practice. Second, in the context of distillation, our emphasis is on capturing the behavior of the agent under a specific policy, and local metrics provide more informative insights in this regard. Finally, local metrics are more interpretable as they provide information about the behavior of the agent in states that are likely to be visited under the policy of interest and ignore the irrelevant parts of the state-action space.

Precisely, we consider local losses defined as the expected value of the aforementioned distances, where the expectation is taken over the state-action pairs visited by the agent at the limit, according to a stationary distribution $\xi_{\bar{\pi}}$ induced by $\bar{\pi}$ (cf. Chapter 2, Section 1.4). As we mentioned earlier, our attention is particularly drawn to discrete latent spaces (which are amenable to model checking). We henceforth make the following assumptions.

Assumption 2: Existence of a stationary distribution

Given a latent stationary policy $\bar{\pi}$, we assume the access to a stationary distribution $\xi_{\bar{\pi}}$ over the state space of the environment $\mathcal{M}$, when it operates under $\bar{\pi}$.

Assumption 3: Discrete latent space

When dealing with discrete spaces, we assume that they are equipped with the discrete metric $\mathbb{1}_{\neq}$. In the following, we *specifically focus on the case where the latent space is finite*, i.e., $\bar{\mathcal{S}}$ and $\bar{\mathcal{A}}$ are finite sets.

Assumption 4: Perfect encoding of the labels

The embedding function preserves the labels, i.e., for all $s \in \mathcal{S}$ and $\bar{s} \in \bar{\mathcal{S}}$,

$$\phi(s) = \bar{s} \text{ implies } \ell(s) = \bar{\ell}(\bar{s}).$$

Assumption 2 is relatively mild in the context of RL. Notably, in episodic MDPs (which encompass the majority of RL scenarios), a unique stationary distribution is guaranteed to exist (cf. Remark 6). We refer the reader to Chapter 2, Section 2.1 for more details. Assumption 4 is actually a mild requirement as well: we show in Section 2 that the embedding function can trivially be trained to preserve the labels of the original environment.

In the remaining of this section, we let $\bar{\pi}$ be a latent stationary policy, $\xi_{\bar{\pi}}$ be a stationary measure induced by $\bar{\pi}$ in $\mathcal{M}$, and $K_V = 2R_{\max}/(1-\gamma)$ be an upper bound on the absolute value difference of the two models. We formally define the local losses as follows.

Definition 37: Local reward and transition losses

The *local reward* and *transition losses* are respectively defined as

$$L_R = \mathbb{E}_{s, \bar{a} \sim \xi_{\bar{\pi}}} |\mathcal{R}(s, \bar{a}) - \bar{\mathcal{R}}(\phi(s), \bar{a})|, \quad L_P = \mathbb{E}_{s, \bar{a} \sim \xi_{\bar{\pi}}} \mathcal{W}_{\bar{a}}(\phi \mathbf{P}(\cdot | s, \bar{a}), \bar{\mathbf{P}}(\cdot | \phi(s), \bar{a})).$$

In the following subsections, we relate the local losses to the bisimulation pseudometrics through upper bounds. Intuitively, these bounds share the property of driving the bisimulation distances to zero as the local losses approach zero. Our results are summarized in Table 3.1.

1.2 Bisimulation Bounds

Reasoning in the local setting, we establish connections between the local losses and the bisimulation pseudometrics *defined under a given latent stationary policy $\bar{\pi}$* , which remarkably satisfy the following properties (cf. Theorem 21): let $s_1, s_2 \in \mathcal{S}$ and $\varphi \in \text{VLC}(\gamma)$

be a closed VLC formulae with discount factors inferior or equal to γ , then²

$$\begin{aligned} |V^{\bar{\pi}}(s_1, \varphi) - V^{\bar{\pi}}(s_2, \varphi)| &\leq \tilde{d}_{\bar{\pi}}^{\ell}(s_1, s_2) = \sup_{\varphi^* \in \text{VLC}(\gamma)} |V^{\bar{\pi}}(s_1, \varphi^*) - V^{\bar{\pi}}(s_2, \varphi^*)|, \text{ and} \\ |V^{\bar{\pi}}(s_1) - V^{\bar{\pi}}(s_2)| &\leq \tilde{d}_{\bar{\pi}}^R(s_1, s_2). \end{aligned} \quad (3.1)$$

We further aim at reasoning about the bisimulation distance between $\mathcal{M}$ and $\overline{\mathcal{M}}$, so we consider the distance defined over the disjoint union of the original and latent space models (cf. Definition 30). Computing the bisimulation distance is arduous in practice: it embeds a fixed point over the entire state-action space which involves the computation of the Wasserstein distance between the transition functions of every pair of states (comprising both those of the original and latent spaces).

Abstraction Quality Guarantees

Alternatively, we claim that the local losses can be used to evaluate the original and latent models' bisimilarity: *whenever the local losses are small, the average bisimulation distance is guaranteed to be small as well*. Interestingly, this also means that, as the losses decrease to zero, the bisimulation distance between the original and latent models almost surely tends to zero as well. This is formally stated in the following theorem.

Theorem 22: Abstraction quality as average bisimulation bounds

The bisimulation distance between the original and latent models is bounded by the local losses in average:

$$\mathbb{E}_{s \sim \xi_{\bar{\pi}}} \tilde{d}_{\bar{\pi}}^{\ell}(s, \phi(s)) \leq \frac{\gamma L_P}{1 - \gamma}, \quad \mathbb{E}_{s \sim \xi_{\bar{\pi}}} \tilde{d}_{\bar{\pi}}^R(s, \phi(s)) \leq \frac{L_R + \gamma K_V \cdot L_P}{1 - \gamma}.$$

Before proving Theorem 22, we first introduce a technical lemma that will be useful in the subsequent proofs.

Lemma 10: Moving the supremum inside the expectation

Let $\mathcal{X}$ be a measurable space and $P, Q \in \Delta(\mathcal{X})$. Then,

$$\sup_{f: \mathcal{X} \rightarrow \mathbb{R}} \mathbb{E}_{\substack{x \sim P \\ y \sim Q}} [f(x) - f(y)] \leq \mathbb{E}_{\substack{x \sim P \\ y \sim Q}} \left[\sup_{f: \mathcal{X} \rightarrow \mathbb{R}} f(x) - f(y) \right]$$

²recall that $V^{\pi}(\cdot, \varphi)$ denotes the values of the formula φ under π and V^{π} those of the expected return.

	Upper bound	PAC bound	Confidence level	# samples from $\xi_{\bar{\pi}}$
$\mathbb{E}_{s \sim \xi_{\bar{\pi}}} \tilde{d}_{\bar{\pi}}^{\ell}(s, \phi(s))$	$\frac{\gamma \hat{L}_{\mathbf{P}}}{1 - \gamma}$	$\frac{\gamma \hat{L}_{\mathbf{P}}}{1 - \gamma} + \varepsilon$	$1 - \delta$	$\frac{-\gamma^2 \log \delta}{2\varepsilon^2(1 - \gamma)^2}$
$\mathbb{E}_{s \sim \xi_{\bar{\pi}}} \tilde{d}_{\bar{\pi}}^{\mathcal{R}}(s, \phi(s))$	$\frac{L_{\mathcal{R}} + \gamma K_V \cdot L_{\mathbf{P}}}{1 - \gamma}$	$\frac{\hat{L}_{\mathcal{R}} + \gamma K_V \cdot \hat{L}_{\mathbf{P}}}{1 - \gamma} + \varepsilon$	$1 - \delta$	$\frac{-K_{\mathcal{R}}(1 + \gamma K_V)^2 \log(\delta/2)}{2\varepsilon^2(1 - \gamma)^2}$
$\forall s \tilde{d}_{\bar{\pi}}^{\ell}(s, \phi(s))$	/	ε	$1 - \frac{\gamma \hat{L}_{\mathbf{P}}}{\varepsilon(1 - \gamma)}$	/
$\forall s \tilde{d}_{\bar{\pi}}^{\mathcal{R}}(s, \phi(s))$	/	ε	$1 - \frac{L_{\mathcal{R}} + \gamma K_V \cdot L_{\mathbf{P}}}{\varepsilon(1 - \gamma)}$	/
$\tilde{d}_{\bar{\pi}}^{\ell}(s_I, \bar{s}_I)$	$\frac{\mathbb{E}_{\bar{\pi}}[\mathbf{T}] \cdot L_{\mathbf{P}}}{1 - \gamma}$	$\frac{\hat{L}_{\mathbf{P}}}{\hat{\xi}_{\text{reset}}(1 - \gamma)} + \varepsilon$	$1 - \delta$	$\frac{-\kappa_1^2 \cdot \log(\delta/2)}{2\hat{\xi}_{\text{reset}}^4 \varepsilon^2(1 - \gamma)^2}$
$\tilde{d}_{\bar{\pi}}^{\mathcal{R}}(s_I, \bar{s}_I)$	$\mathbb{E}_{\bar{\pi}}[\mathbf{T}] \frac{L_{\mathcal{R}}/\gamma + K_V L_{\mathbf{P}}}{1 - \gamma}$	$\frac{\hat{L}_{\mathcal{R}}/\gamma + K_V \hat{L}_{\mathbf{P}}}{\hat{\xi}_{\text{reset}}(1 - \gamma)} + \varepsilon$	$1 - \delta$	$\frac{-K_{\mathcal{R}} \cdot \kappa_2^2 \cdot \log(\delta/3)}{2\gamma^2 \hat{\xi}_{\text{reset}}^4 \varepsilon^2(1 - \gamma)^2}$
$\tilde{d}_{\bar{\pi}}^{\ell}(s_1, s_2)$	$\frac{\gamma \hat{L}_{\mathbf{P}}}{1 - \gamma} \cdot f_{\bar{\pi}}(s_1, s_2)$	/	/	/
$\tilde{d}_{\bar{\pi}}^{\mathcal{R}}(s_1, s_2)$	$\frac{L_{\mathcal{R}} + \gamma K_V \cdot L_{\mathbf{P}}}{1 - \gamma} \cdot f_{\bar{\pi}}(s_1, s_2)$	/	/	/
$\forall s_1, s_2 \text{ such that } \phi(s_1) = \phi(s_2)$				

Table 3.1: Upper bounds derived from the local losses on bisimulation pseudometrics. Additionally, we provide PAC upper bounds given an error term $\varepsilon > 0$, confidence level, and a sufficient number of samples from $\xi_{\bar{\pi}}$ to achieve that level. The confidence may depend on a parameter $\delta \in (0, 1)$. $\hat{L}_{\mathcal{R}}, \hat{L}_{\mathbf{P}}$ represent approximations of the local losses and $\hat{\xi}_{\text{reset}}$ of $\xi_{\bar{\pi}}(s_{\text{reset}})$ (Lemma 14). $\mathbb{E}_{\bar{\pi}}[\mathbf{T}]$ is the *average episode length* when $\bar{\pi}$ is executed. We let $K_{\mathcal{R}} = \max\{4R_{\text{MAX}}^2, 1\}$, $f_{\bar{\pi}}(s_1, s_2) = \left(\frac{1}{\xi_{\bar{\pi}}(s_1)} + \frac{1}{\xi_{\bar{\pi}}(s_2)}\right)$, $\kappa_1 = \hat{L}_{\mathbf{P}} + \hat{\xi}_{\text{reset}}(1 + \varepsilon(1 - \gamma))$, and $\kappa_2 = \hat{L}_{\mathcal{R}} + \gamma K_V \hat{L}_{\mathbf{P}} + \hat{\xi}_{\text{reset}}(1 + \gamma K_V + \gamma \varepsilon(1 - \gamma))$.

Proof. For all functions $f : \mathcal{X} \rightarrow \mathbb{R}$ and $x, y \in \mathcal{X}$, $f(x) - f(y) \leq \sup_f f(x) - f(y)$, meaning that $\mathbb{E}[f(x) - f(y)] \leq \mathbb{E}[\sup_f f(x) - f(y)]$. In particular, this also holds for $f^* = \arg \sup_f \mathbb{E}[f(x) - f(y)]$, i.e., $\mathbb{E}[f^*(x) - f^*(y)] \leq \mathbb{E}[\sup_f f(x) - f(y)]$. $\square$

Proof (of Theorem 22). Recall that $\tilde{d}_{\bar{\pi}}^{\ell}$ and $\tilde{d}_{\bar{\pi}}^{\mathcal{R}}$ are respectively one- and K_V -bounded distance pseudometrics by definition (Theorem 20). By assumption, the latent space is finite, meaning it is equipped with the discrete metric $\bar{d} = \mathbb{1}_{\neq}$ (Assumption 3). Therefore, for all $\bar{s}_1, \bar{s}_2 \in \bar{\mathcal{S}}$, we have

$$\tilde{d}_{\bar{\pi}}^{\ell}(\bar{s}_1, \bar{s}_2) \leq \bar{d}(\bar{s}_1, \bar{s}_2), \quad \tilde{d}_{\bar{\pi}}^{\mathcal{R}}(\bar{s}_1, \bar{s}_2) \leq K_V \cdot \bar{d}(\bar{s}_1, \bar{s}_2).$$

Let $\clubsuit \in \{\ell, \mathcal{R}\}$ and $K_{\clubsuit} = \begin{cases} 1 & \text{if } \clubsuit = \ell \\ K_V & \text{otherwise, when } \clubsuit = \mathcal{R}; \end{cases}$ then, we have

$$\begin{aligned}
& \mathbb{E}_{s \sim \xi_{\bar{\pi}}} \mathcal{W}_{\tilde{d}_{\bar{\pi}}^{\clubsuit}}(\phi \mathbf{P}_{\bar{\pi}}(\cdot | s), \bar{\mathbf{P}}_{\bar{\pi}}(\cdot | \phi(s))) \\
&= \mathbb{E}_{s \sim \xi_{\bar{\pi}}} \left[\inf_{\omega \in \Pi(\phi \mathbf{P}_{\bar{\pi}}(\cdot | s), \bar{\mathbf{P}}_{\bar{\pi}}(\cdot | \phi(s)))} \sum_{\bar{s}_1, \bar{s}_2 \in \bar{\mathcal{S}}} \tilde{d}_{\bar{\pi}}^{\clubsuit}(\bar{s}_1, \bar{s}_2) \omega(\bar{s}_1, \bar{s}_2) \right] \\
&\leq \mathbb{E}_{s \sim \xi_{\bar{\pi}}} \left[\inf_{\omega \in \Pi(\phi \mathbf{P}_{\bar{\pi}}(\cdot | s), \bar{\mathbf{P}}_{\bar{\pi}}(\cdot | \phi(s)))} \sum_{\bar{s}_1, \bar{s}_2 \in \bar{\mathcal{S}}} K_{\clubsuit} \mathbb{1}_{\neq}(\bar{s}_1, \bar{s}_2) \omega(\bar{s}_1, \bar{s}_2) \right] \quad (\tilde{d}_{\bar{\pi}}^{\clubsuit} \text{ is } K_{\clubsuit}\text{-bounded in } \bar{\mathcal{S}}) \\
&= K_{\clubsuit} \mathbb{E}_{s \sim \xi_{\bar{\pi}}} \left[\inf_{\omega \in \Pi(\phi \mathbf{P}_{\bar{\pi}}(\cdot | s), \bar{\mathbf{P}}_{\bar{\pi}}(\cdot | \phi(s)))} \sum_{\bar{s}_1 \neq \bar{s}_2} \omega(\bar{s}_1, \bar{s}_2) \right] \\
&= K_{\clubsuit} \mathbb{E}_{s \sim \xi_{\bar{\pi}}} d_{TV}(\phi \mathbf{P}_{\bar{\pi}}(\cdot | s), \bar{\mathbf{P}}_{\bar{\pi}}(\cdot | \phi(s))) \quad (\text{TV coincides with } \mathcal{W}_{\mathbb{1}_{\neq}}; \text{ see Definition 34}) \\
&= K_{\clubsuit} \mathbb{E}_{s \sim \xi_{\bar{\pi}}} \left[\frac{1}{2} \sum_{\bar{s}' \in \bar{\mathcal{S}}} \left| \phi \mathbf{P}_{\bar{\pi}}(\bar{s}' | s) - \bar{\mathbf{P}}_{\bar{\pi}}(\bar{s}' | \phi(s)) \right| \right] \\
&= K_{\clubsuit} \mathbb{E}_{s \sim \xi_{\bar{\pi}}} \left[\frac{1}{2} \sum_{\bar{a} \sim \bar{\pi}(\cdot | s)} \left| \mathbb{E}_{\phi \mathbf{P}(\bar{s}' | s, \bar{a})} - \mathbb{E}_{\bar{\mathbf{P}}(\bar{s}' | \phi(s), \bar{a})} \right| \right] \\
&= K_{\clubsuit} \mathbb{E}_{s \sim \xi_{\bar{\pi}}} \left[\frac{1}{2} \sum_{\bar{a} \sim \bar{\pi}(\cdot | s)} \left| \mathbb{E}_{\phi \mathbf{P}(\bar{s}' | s, \bar{a})} - \mathbb{E}_{\bar{\mathbf{P}}(\bar{s}' | \phi(s), \bar{a})} \right| \right] \\
&\leq K_{\clubsuit} \mathbb{E}_{s \sim \xi_{\bar{\pi}}} \mathbb{E}_{\bar{a} \sim \bar{\pi}(\cdot | s)} \left[\frac{1}{2} \sum_{\bar{s}' \in \bar{\mathcal{S}}} \left| \phi \mathbf{P}(\bar{s}' | s, \bar{a}) - \bar{\mathbf{P}}(\bar{s}' | \phi(s), \bar{a}) \right| \right] \quad (\text{Jensen's inequality}^a) \\
&\leq K_{\clubsuit} L_P.
\end{aligned}$$

$$\begin{aligned}
& \mathbb{E}_{s \sim \xi_{\bar{\pi}}} \tilde{d}_{\bar{\pi}}^{\clubsuit}(s, \phi(s)) \\
&= \mathbb{E}_{s \sim \xi_{\bar{\pi}}} \left[|\mathcal{R}_{\bar{\pi}}(s) - \bar{\mathcal{R}}_{\bar{\pi}}(\phi(s))| \cdot \mathbb{1}_{\{\clubsuit = \mathcal{R}\}} + \gamma \mathcal{W}_{\tilde{d}_{\bar{\pi}}^{\clubsuit}}(\mathbf{P}_{\bar{\pi}}(\cdot | s), \bar{\mathbf{P}}_{\bar{\pi}}(\cdot | \phi(s))) \right] \\
&\quad (\ell(s) = \ell(\phi(s)) \text{ by Assumption 4}) \\
&\leq L_{\mathcal{R}} \cdot \mathbb{1}_{\{\clubsuit = \mathcal{R}\}} + \gamma \cdot \mathbb{E}_{s \sim \xi_{\bar{\pi}}} \left[\mathcal{W}_{\tilde{d}_{\bar{\pi}}^{\clubsuit}}(\mathbf{P}_{\bar{\pi}}(\cdot | s), \bar{\mathbf{P}}_{\bar{\pi}}(\cdot | \phi(s))) \right] \\
&\leq L_{\mathcal{R}} \cdot \mathbb{1}_{\{\clubsuit = \mathcal{R}\}} + \gamma \cdot \mathbb{E}_{s \sim \xi_{\bar{\pi}}} \left[\mathcal{W}_{\tilde{d}_{\bar{\pi}}^{\clubsuit}}(\mathbf{P}_{\bar{\pi}}(\cdot | s), \phi \mathbf{P}_{\bar{\pi}}(\cdot | s)) + \mathcal{W}_{\tilde{d}_{\bar{\pi}}^{\clubsuit}}(\phi \mathbf{P}_{\bar{\pi}}(\cdot | s), \bar{\mathbf{P}}_{\bar{\pi}}(\cdot | \phi(s))) \right] \\
&\quad (\mathcal{W}_{\tilde{d}_{\bar{\pi}}^{\clubsuit}} \text{ is a pseudometric, so we apply the triangle inequality})
\end{aligned}$$

$$\begin{aligned}
 &\leq L_{\mathcal{R}} \cdot \mathbb{1}\{\clubsuit = \mathcal{R}\} + \gamma K_{\clubsuit} \cdot L_P + \gamma \cdot \mathbb{E}_{s \sim \xi_{\pi}} \mathcal{W}_{\tilde{d}_{\pi}^{\clubsuit}}(\mathbf{P}_{\pi}(\cdot | s), \phi \mathbf{P}_{\pi}(\cdot | s)) \\
 &\hspace{15em} \text{(by the inequality derived above)} \\
 &\leq L_{\mathcal{R}} \cdot \mathbb{1}\{\clubsuit = \mathcal{R}\} + \gamma K_{\clubsuit} \cdot L_P + \gamma \cdot \mathbb{E}_{s \sim \xi_{\pi}} \tilde{d}_{\pi}^{\clubsuit}(s, \phi(s)).
 \end{aligned}$$

To see how we pass from the penultimate to the last line, recall that

$$\mathbb{E}_{s \sim \xi_{\pi}} \mathcal{W}_{\tilde{d}_{\pi}^{\clubsuit}}(\mathbf{P}_{\pi}(\cdot | s), \phi \mathbf{P}_{\pi}(\cdot | s)) = \mathbb{E}_{s \sim \xi_{\pi}} \left[\sup_{f \in \text{Lipsch}_{\tilde{d}_{\pi}^{\clubsuit}}^1} \mathbb{E}_{s' \sim \mathbf{P}_{\pi}(\cdot | s)} f(s') - \mathbb{E}_{\bar{s}' \sim \phi \mathbf{P}_{\pi}(\cdot | s)} f(\bar{s}') \right],$$

where

$$\text{Lipsch}_{\tilde{d}_{\pi}^{\clubsuit}}^1 = \{f : \mathcal{S} \cup \bar{\mathcal{S}} \rightarrow \mathbb{R} \mid f(s_1^*) - f(s_2^*) \leq \tilde{d}_{\pi}^{\clubsuit}(s_1^*, s_2^*) \text{ for all } s_1^*, s_2^* \in \mathcal{S} \cup \bar{\mathcal{S}}\}.$$

Then, we have

$$\begin{aligned}
 &\mathbb{E}_{s \sim \xi_{\pi}} \left[\sup_{f \in \text{Lipsch}_{\tilde{d}_{\pi}^{\clubsuit}}^1} \mathbb{E}_{s' \sim \mathbf{P}_{\pi}(\cdot | s)} f(s') - \mathbb{E}_{\bar{s}' \sim \phi \mathbf{P}_{\pi}(\cdot | s)} f(\bar{s}') \right] \\
 &= \mathbb{E}_{s \sim \xi_{\pi}} \left[\sup_{f \in \text{Lipsch}_{\tilde{d}_{\pi}^{\clubsuit}}^1} \mathbb{E}_{s' \sim \mathbf{P}_{\pi}(\cdot | s)} f(s') - \mathbb{E}_{s' \sim \mathbf{P}_{\pi}(\cdot | s)} f(\phi(s')) \right] \\
 &= \mathbb{E}_{s \sim \xi_{\pi}} \left[\sup_{f \in \text{Lipsch}_{\tilde{d}_{\pi}^{\clubsuit}}^1} \mathbb{E}_{s' \sim \mathbf{P}_{\pi}(\cdot | s)} [f(s') - f(\phi(s'))] \right] \\
 &\leq \mathbb{E}_{s \sim \xi_{\pi}} \mathbb{E}_{s' \sim \mathbf{P}_{\pi}(\cdot | s)} \left[\sup_{f \in \text{Lipsch}_{\tilde{d}_{\pi}^{\clubsuit}}^1} f(s') - f(\phi(s')) \right] \hspace{10em} \text{(by Lemma 10)} \\
 &= \mathbb{E}_{s \sim \xi_{\pi}} \sup_{f \in \text{Lipsch}_{\tilde{d}_{\pi}^{\clubsuit}}^1} f(s) - f(\phi(s)) \hspace{10em} \text{(by definition of the stationary measure } \xi_{\pi}) \\
 &\leq \mathbb{E}_{s \sim \xi_{\pi}} \tilde{d}_{\pi}^{\clubsuit}(s, \phi(s)). \hspace{15em} \text{(since } f \in \text{Lipsch}_{\tilde{d}_{\pi}^{\clubsuit}}^1)
 \end{aligned}$$

Finally, we have

$$\begin{aligned}
 (1 - \gamma) \cdot \mathbb{E}_{s \sim \xi_{\pi}} \tilde{d}_{\pi}^{\clubsuit}(s, \phi(s)) &\leq L_{\mathcal{R}} \cdot \mathbb{1}\{\clubsuit = \mathcal{R}\} + \gamma K_{\clubsuit} \cdot L_P \\
 \iff \mathbb{E}_{s \sim \xi_{\pi}} \tilde{d}_{\pi}^{\clubsuit}(s, \phi(s)) &\leq \frac{L_{\mathcal{R}} \cdot \mathbb{1}\{\clubsuit = \mathcal{R}\} + \gamma K_{\clubsuit} \cdot L_P}{1 - \gamma}.
 \end{aligned}$$

□

^aLet X be a random variable and f be a convex function. Then, $f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)]$ [Jensen, 1906].

Since the value function is Lipschitz continuous with respect to the bisimulation distance (Equation 3.1) and since we consider pseudometrics defined over the disjoint union of the original and latent models (Definition 30), we obtain as a direct corollary of Theorem 22 the following result, where the one for the expected return corresponds to the bounds established by Gelada et al. [2019] in the context of DeepMDPs (Remark 17).

Corollary 2: Abstraction quality as value difference bounds

Let $s \in S$ and $\phi \in \text{VLC}(\gamma)$ be a closed VLC formula with discount factors inferior or equal to γ . Then, *the average value gap between the original and latent models is bounded by the local losses:*

$$\mathbb{E}_{s \sim \xi_{\bar{\pi}}} |V^{\bar{\pi}}(s, \phi) - \bar{V}^{\bar{\pi}}(\phi(s), \phi)| \leq \frac{\gamma L_P}{1 - \gamma}, \quad \mathbb{E}_{s \sim \xi_{\bar{\pi}}} |V^{\bar{\pi}}(s) - \bar{V}^{\bar{\pi}}(\phi(s))| \leq \frac{L_R + \gamma K_V \cdot L_P}{1 - \gamma}.$$

Episodic setting. In the context of RL, training environments are often episodic (refer to Chapter 2 Section 2.1 for a comprehensive discussion). We showed that the episodic return does not necessarily align with the usual expected return (Lemma 4). Hence, a natural question to consider is how we interpret the bounds on value differences within the context of episodic MDPs. For simplicity, we make the following (straightforward) assumption.

Assumption 5: Reset label

Whenever an MDP is episodic, we assume the existence of an atomic proposition **reset** $\in \text{AP}$ so that **reset** $\in \ell(s)$ if and only if $s \in S$ is the reset state, i.e., $s = s_{\text{reset}}$.

Combined with Assumption 4, this assumption basically implies the encoding of the reset state of the original MDP into the reset state of the latent MDP, supposing both models are episodic. Then, the value difference bound holds for the episodic return as well.

Lemma 11: Episodic formulation of the value difference bound *

Assume that $\mathcal{M}$ and $\bar{\mathcal{M}}$ are episodic processes and let $s \in S$. *The average episodic value gap between the original and latent models is bounded by the local losses:*

$$\mathbb{E}_{s \sim \xi_{\bar{\pi}}} |V_{\text{epis}}^{\bar{\pi}}(s) - \bar{V}_{\text{epis}}^{\bar{\pi}}(\phi(s))| \leq \frac{L_R + \gamma K_V \cdot L_P}{1 - \gamma}.$$

Proof. First, notice that Assumptions 4 and 5 imply that $\phi(s_{\text{reset}})$ is the reset state of $\bar{\mathcal{M}}$. Then, $|V_{\text{epis}}^{\bar{\pi}}(s_{\text{reset}}) - \bar{V}_{\text{epis}}^{\bar{\pi}}(\phi(s_{\text{reset}}))| = 0$ by definition of the episodic return (Definition 18). Let us further substitute each value function with its definition inside

the absolute value difference:

$$\begin{aligned}
 & \mathbb{E}_{s \sim \xi_\pi} \left| V_{\text{epis}}^\pi(s) - \bar{V}_{\text{epis}}^\pi(\phi(s)) \right| \\
 &= \mathbb{E}_{s, \bar{a} \sim \xi_\pi} \left[\mathbb{1} \{s \neq s_{\text{reset}}\} \cdot \left| \mathcal{R}(s, \bar{a}) + \mathbb{E}_{s' \sim \mathbf{P}(\cdot|s, \bar{a})} [\gamma V_{\text{epis}}^\pi(s')] - \bar{\mathcal{R}}(\phi(s), \bar{a}) - \mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot|\phi(s), \bar{a})} [\gamma \bar{V}_{\text{epis}}^\pi(\bar{s}')] \right| \right] \\
 & \quad \text{(since } |V_{\text{epis}}^\pi(s) - \bar{V}_{\text{epis}}^\pi(\phi(s))| = 0 \text{ whenever } s = s_{\text{reset}} \text{)} \\
 &\leq \mathbb{E}_{s, \bar{a} \sim \xi_\pi} \left| \mathcal{R}(s, \bar{a}) + \mathbb{E}_{s' \sim \mathbf{P}(\cdot|s, \bar{a})} [\gamma V_{\text{epis}}^\pi(s')] - \bar{\mathcal{R}}(\phi(s), \bar{a}) - \mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot|\phi(s), \bar{a})} [\gamma \bar{V}_{\text{epis}}^\pi(\bar{s}')] \right|.
 \end{aligned}$$

The rest of the proof is similar to the one establishing the value difference bound of Gelada et al. [2019, Lemma 8] that we adapt here in the context of episodic return and discrete latent models, for the sake of completeness:

$$\begin{aligned}
 & \mathbb{E}_{s, \bar{a} \sim \xi_\pi} \left| \mathcal{R}(s, \bar{a}) + \mathbb{E}_{s' \sim \mathbf{P}(\cdot|s, \bar{a})} [\gamma V_{\text{epis}}^\pi(s')] - \bar{\mathcal{R}}(\phi(s), \bar{a}) - \mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot|\phi(s), \bar{a})} [\gamma \bar{V}_{\text{epis}}^\pi(\bar{s}')] \right| \\
 &\leq \mathbb{E}_{s, \bar{a} \sim \xi_\pi} \left| \mathcal{R}(s, \bar{a}) - \bar{\mathcal{R}}(\phi(s), \bar{a}) \right| + \gamma \mathbb{E}_{s, \bar{a} \sim \xi_\pi} \left| \mathbb{E}_{s' \sim \mathbf{P}(\cdot|s, \bar{a})} [V_{\text{epis}}^\pi(s')] - \mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot|\phi(s), \bar{a})} [\bar{V}_{\text{epis}}^\pi(\bar{s}')] \right| \\
 & \quad \text{(triangle inequality)} \\
 &= L_R + \gamma \mathbb{E}_{s, \bar{a} \sim \xi_\pi} \left| \mathbb{E}_{s' \sim \mathbf{P}(\cdot|s, \bar{a})} [V_{\text{epis}}^\pi(s')] - \mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot|\phi(s), \bar{a})} [\bar{V}_{\text{epis}}^\pi(\bar{s}')] \right| \quad \text{(by definition of } L_R \text{)} \\
 &\leq L_R + \gamma \mathbb{E}_{s, \bar{a} \sim \xi_\pi} \left| \mathbb{E}_{s' \sim \mathbf{P}(\cdot|s, \bar{a})} [V_{\text{epis}}^\pi(s') - \bar{V}_{\text{epis}}^\pi(\phi(s'))] + \mathbb{E}_{\substack{\bar{s}' \sim \bar{\mathbf{P}}(\cdot|\phi(s), \bar{a}) \\ s' \sim \mathbf{P}(\cdot|s, \bar{a})}} [\bar{V}_{\text{epis}}^\pi(\phi(s')) - \bar{V}_{\text{epis}}^\pi(\bar{s}')] \right| \\
 &\leq L_R + \gamma \mathbb{E}_{s, \bar{a} \sim \xi_\pi} \left| \mathbb{E}_{s' \sim \mathbf{P}(\cdot|s, \bar{a})} [V_{\text{epis}}^\pi(s') - \bar{V}_{\text{epis}}^\pi(\phi(s'))] \right| \\
 & \quad + \gamma \mathbb{E}_{s, \bar{a} \sim \xi_\pi} \left| \mathbb{E}_{\substack{\bar{s}' \sim \bar{\mathbf{P}}(\cdot|\phi(s), \bar{a}) \\ s' \sim \mathbf{P}(\cdot|s, \bar{a})}} [\bar{V}_{\text{epis}}^\pi(\phi(s')) - \bar{V}_{\text{epis}}^\pi(\bar{s}')] \right| \quad \text{(triangle inequality)} \\
 &= L_R + \gamma \mathbb{E}_{s, \bar{a} \sim \xi_\pi} \left| \mathbb{E}_{s' \sim \mathbf{P}(\cdot|s, \bar{a})} [V_{\text{epis}}^\pi(s') - \bar{V}_{\text{epis}}^\pi(\phi(s'))] \right| \\
 & \quad + \gamma \mathbb{E}_{s, \bar{a} \sim \xi_\pi} \left| \mathbb{E}_{s' \sim \mathbf{P}(\cdot|s, \bar{a})} \bar{V}_{\text{epis}}^\pi(\phi(s')) - \mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot|\phi(s), \bar{a})} \bar{V}_{\text{epis}}^\pi(\bar{s}') \right|.
 \end{aligned}$$

Notice that since $\bar{\mathcal{R}}$ is bounded by R_{MAX} , $\left| \bar{V}_{\text{epis}}^{\bar{\pi}} \right|$ is bounded by $R_{\text{MAX}}/1-\gamma = K_V/2$. Take $f = 2\bar{V}_{\text{epis}}^{\bar{\pi}}/K_V$, it is clear that f is one-bounded. Then, $\mathbb{E}_{s,\bar{a}} \left| \mathbb{E}_{s'} \bar{V}_{\text{epis}}^{\bar{\pi}}(\phi(s')) - \mathbb{E}_{\bar{s}'} \bar{V}_{\text{epis}}^{\bar{\pi}}(\bar{s}') \right| = \mathbb{E}_{s,\bar{a}} [K_V/2 \cdot |\mathbb{E}_{s'} f(\phi(s')) - \mathbb{E}_{\bar{s}'} f(\bar{s}')|] \leq K_V \cdot \mathbb{E}_{s,\bar{a}} \mathcal{W}_{\mathbb{I}_{\neq}}(\phi\mathbf{P}(s,\bar{a}), \bar{\mathbf{P}}(\phi(s),\bar{a}))$ by seeing d_{TV} as an integral probability metric (Definition 34). This yields

$$\begin{aligned}
& \mathbb{E}_{s \sim \xi_{\bar{\pi}}} \left| V_{\text{epis}}^{\bar{\pi}}(s) - \bar{V}_{\text{epis}}^{\bar{\pi}}(\phi(s)) \right| \\
& \leq L_{\mathcal{R}} + \gamma \mathbb{E}_{s,\bar{a} \sim \xi_{\bar{\pi}}} \left| \mathbb{E}_{s' \sim \mathbf{P}(\cdot|s,\bar{a})} \left[V_{\text{epis}}^{\bar{\pi}}(s') - \bar{V}_{\text{epis}}^{\bar{\pi}}(\phi(s')) \right] \right| + \gamma K_V \cdot \mathbb{E}_{s,\bar{a} \sim \xi_{\bar{\pi}}} \mathcal{W}_{\bar{d}}(\phi\mathbf{P}(\cdot|s,\bar{a}), \bar{\mathbf{P}}(\cdot|\phi(s),\bar{a})) \\
& \leq L_{\mathcal{R}} + \gamma \mathbb{E}_{s,\bar{a} \sim \xi_{\bar{\pi}}} \left| \mathbb{E}_{s' \sim \mathbf{P}(\cdot|s,\bar{a})} \left[V_{\text{epis}}^{\bar{\pi}}(s') - \bar{V}_{\text{epis}}^{\bar{\pi}}(\phi(s')) \right] \right| + \gamma K_V \cdot L_{\mathbf{P}} \quad (\text{by definition of } L_{\mathbf{P}}) \\
& \leq L_{\mathcal{R}} + \gamma \mathbb{E}_{s,\bar{a} \sim \xi_{\bar{\pi}}} \mathbb{E}_{s' \sim \mathbf{P}(\cdot|s,\bar{a})} \left| V_{\text{epis}}^{\bar{\pi}}(s') - \bar{V}_{\text{epis}}^{\bar{\pi}}(\phi(s')) \right| + \gamma K_V \cdot L_{\mathbf{P}} \quad (\text{Jensen's inequality}) \\
& = L_{\mathcal{R}} + \gamma \mathbb{E}_{s \sim \xi_{\bar{\pi}}} \left| V_{\text{epis}}^{\bar{\pi}}(s) - \bar{V}_{\text{epis}}^{\bar{\pi}}(\phi(s)) \right| + \gamma K_V \cdot L_{\mathbf{P}}. \\
& \hspace{15em} (\text{by definition of the stationary measure } \xi_{\bar{\pi}})
\end{aligned}$$

Then, we finally obtain

$$\begin{aligned}
(1-\gamma) \mathbb{E}_{s \sim \xi_{\bar{\pi}}} \left| V_{\text{epis}}^{\bar{\pi}}(s) - \bar{V}_{\text{epis}}^{\bar{\pi}}(\phi(s)) \right| & \leq L_{\mathcal{R}} + \gamma K_V \cdot L_{\mathbf{P}} \\
\iff \mathbb{E}_{s \sim \xi_{\bar{\pi}}} \left| V_{\text{epis}}^{\bar{\pi}}(s) - \bar{V}_{\text{epis}}^{\bar{\pi}}(\phi(s)) \right| & \leq \frac{L_{\mathcal{R}} + \gamma K_V \cdot L_{\mathbf{P}}}{1-\gamma}.
\end{aligned}$$

□

Above and beyond expectations. All the bounds presented so far are formalized as expectations. This might be a bit inconvenient in practice, especially if one desires to know the value difference for specific states. Using plain Markov's inequality³, one can derive PAC bounds on the bisimulation distances and value differences, removing the need of formalizing them through expectations. Consequently, as $L_{\mathcal{R}}$ and $L_{\mathbf{P}}$ tend towards zero, the upcoming bounds highlight the convergence in probability of the bisimulation distance as the local losses decrease: the probability of error below ε approaches one, and that for all $\varepsilon > 0$.

³also referred to as the Chebyshev's inequality [Stein and Shakarchi, 2005]

Lemma 12: PAC abstraction quality bounds *

Let $\varepsilon \in (0, 1)$ be an error term. Given any $K_1, K_2 > 0$, define

$$\delta(K_1, K_2) = \frac{K_1 \cdot L_R + \gamma K_2 \cdot L_P}{\varepsilon(1 - \gamma)}.$$

Then, each of the following holds with probability^a at least $1 - \delta(K_1, K_2)$: for all states $s \in S$,

- (i) $\tilde{d}_\pi^\ell(s, \phi(s)) < \varepsilon$ with $K_1 = 0$ and $K_2 = 1$;
- (ii) $\tilde{d}_\pi^R(s, \phi(s)) < \varepsilon$ with $K_1 = 1$ and $K_2 = K_V$;
- (iii) given any $\varphi \in \text{VLC}(\gamma)$, $|V^\pi(s, \varphi) - \bar{V}^\pi(\phi(s), \varphi)| < \varepsilon$ with $K_1 = 0$ and $K_2 = 1$;
- (iv) $|V^\pi(s) - \bar{V}^\pi(\phi(s))| < \varepsilon$ with $K_1 = 1$ and $K_2 = K_V$; and
- (v) $|V_{\text{epis}}^\pi(s) - \bar{V}_{\text{epis}}^\pi(\phi(s))| < \varepsilon$ with $K_1 = 1$ and $K_2 = K_V$, assuming both the original and latent models are episodic.

^awith respect to the measure ξ_π .

Proof. Let X be a random variable over S so that $X(s)$ takes values in

$$\begin{aligned} & \left\{ \tilde{d}_\pi^\ell(s, \phi(s)), |V^\pi(s, \varphi) - \bar{V}^\pi(\phi(s), \varphi)| \right\} \quad \text{with } K_1 = 0 \text{ and } K_2 = 1, \text{ or} \\ & \left\{ \tilde{d}_\pi^R(s, \phi(s)), |V^\pi(s) - \bar{V}^\pi(\phi(s))|, |V_{\text{epis}}^\pi(s) - \bar{V}_{\text{epis}}^\pi(\phi(s))| \right\} \quad \text{with } K_1 = 1 \text{ and } K_2 = K_V. \end{aligned}$$

We have by Theorem 22, Corollary 2, and Lemma 11 that

$$\mathbb{E}_{s \sim \xi_\pi} [X] \leq \frac{K_1 \cdot L_R + \gamma K_2 \cdot L_P}{1 - \gamma}.$$

Then, by the Markov's inequality [Stein and Shakarchi, 2005], we have

$$\xi_\pi(\{s \in S \mid X(s) \geq \varepsilon\}) \leq \frac{\mathbb{E}_{s \sim \xi_\pi} [X]}{\varepsilon} \leq \frac{K_1 \cdot L_R + \gamma K_2 \cdot L_P}{\varepsilon(1 - \gamma)} = \delta(K_1, K_2).$$

Therefore, $\xi_\pi(\{s \in S \mid X(s) < \varepsilon\}) = 1 - \xi_\pi(\{s \in S \mid X(s) \geq \varepsilon\}) \geq 1 - \delta(K_1, K_2)$. $\square$

In the context of episodic MDPs, a more transparent bound can be derived without introducing a confidence level on the error. Recall that the bisimulation between two MDPs corresponds to the bisimulation over the disjoint union of their state spaces between their initial states (cf. Definition 30). In the following, we thus formalize our

bounds in terms of the initial states of the two models. Intuitively, the result follows from substituting the stationary distribution into the bounds of Theorem 22 and observing that $1/\xi_{\bar{\pi}}(s_{\text{reset}})$ corresponds to the *average episode length* (AEL), i.e.,

$$\frac{1}{\xi_{\bar{\pi}}(s_{\text{reset}})} = \mathbb{E}_{\bar{\pi}}^{\mathcal{M}} \left[\sum_{T=0}^{\infty} (T+1) \cdot \mathbb{1} \left\{ \langle s_{0:T}, a_{0:T-1} \rangle \in \text{Episodes}_{\mathcal{M}} \right\} \right].$$

Indeed, when $\mathcal{M}$ is episodic, it is irreducible and recurrent (cf. Chapter 2, Sections 1.4 and 2.1). Then, given the random variable

$$\mathbf{T}_s(\langle s_{0:\infty}, a_{0:\infty} \rangle) = \sum_{T=1}^{\infty} T \cdot \mathbb{1} \{s_T = s \text{ and } s_t \neq s \text{ for all } 0 < t < T\}, \quad (3.2)$$

we have $\xi_{\pi}(s) = 1/\mathbb{E}_{\pi}[\mathbf{T}_s | s_I = s]$ for any $s \in \mathcal{S}$ and stationary policy π , where $\mathbb{E}_{\pi}[\mathbf{T}_s | s_I = s]$ is the *mean recurrence time* of s under π [Serfoso, 2009, Chapter 1, Theorem 54].

To derive abstraction quality bounds specifically from the initial states, one simply needs to multiply the bounds derived in this section by the AEL constant. Then, the main property of our bounds is now *surely*⁴ satisfied: whenever the losses approach zero, the bisimulation distance and value tends towards zero as well.

Theorem 23: Abstraction quality from the initial states *

Assume that $\mathcal{M}$ and $\bar{\mathcal{M}}$ are episodic processes and let $\mathbb{E}_{\bar{\pi}}[\mathbf{T}] = 1/\xi_{\bar{\pi}}(s_{\text{reset}})$ denote the *average episode length* of $\mathcal{M}$ under $\bar{\pi}$. The bisimulation distance between the original and latent models at their initial states is bounded by the local losses:

$$\begin{aligned} \tilde{d}_{\bar{\pi}}^L(s_I, \bar{s}_I) &\leq \frac{\mathbb{E}_{\bar{\pi}}[\mathbf{T}] \cdot L_P}{1 - \gamma}, & \tilde{d}_{\bar{\pi}}^R(s_I, \bar{s}_I) &\leq \mathbb{E}_{\bar{\pi}}[\mathbf{T}] \frac{L_R/\gamma + K_V \cdot L_P}{1 - \gamma}, \\ |V^{\bar{\pi}}(s_I, \varphi) - \bar{V}^{\bar{\pi}}(\bar{s}_I, \varphi)| &\leq \frac{\mathbb{E}_{\bar{\pi}}[\mathbf{T}] \cdot L_P}{1 - \gamma}, & |V_{\text{epis}}^{\bar{\pi}}(s_I) - \bar{V}_{\text{epis}}^{\bar{\pi}}(\bar{s}_I)| &\leq \mathbb{E}_{\bar{\pi}}[\mathbf{T}] \frac{L_R/\gamma + K_V \cdot L_P}{1 - \gamma}, \end{aligned}$$

where $\varphi \in \text{VLC}(\gamma)$.

Before proving the theorem, we first state a technical lemma that will be useful in the subsequent proofs.

⁴in contrast to *almost surely*, with Lemma 12.

Lemma 13: Pointwise bound on a random variable by its scaled expectation

Let $P \in \Delta(S)$ be a probability distribution over S and X be a *positive* random variable over S . Then, for any *measurable state* $s \in S$ so that $\{s\} \in \Sigma(S)$, we have

$$X(s) \leq \frac{1}{P(s)} \cdot \mathbb{E}_{s \sim P} [X].$$

Proof. $\frac{\mathbb{E}[X]}{P(s)} = \int_S \frac{X(s')}{P(s)} dP(s') = X(s) + \int_{S \setminus \{s\}} \frac{X(s')}{P(s)} dP(s') \geq X(s).$ $\square$

Proof (of Theorem 23). First, recall that we assumed that the inverse labeling function of every $A \subseteq \mathbf{AP}$ is measurable (cf. the discussion after Definition 5), i.e., $\ell^{-1}(A) \in \Sigma(S)$. Then, the reset state is measurable by Assumption 5: s_{reset} is the only state such that **reset** $\in \ell(s_{reset})$, so $\{s_{reset}\} = \ell^{-1}(\{\mathbf{reset}\}) \in \Sigma(S)$.

Second, recall that the reset state of any episodic process deterministically transitions to the initial state (cf. Definition 16), i.e., $\mathbf{P}_{\bar{\pi}}(s_I | s_{reset}) = \bar{\mathbf{P}}_{\bar{\pi}}(\bar{s}_I | \phi(s_{reset})) = 1$. Let $\clubsuit = \{\ell, \mathcal{R}\}$, since ϕ preserves the labeling function (Assumption 4), we have $\ell(s_{reset}) = \ell(\phi(s_{reset}))$ and

$$\tilde{d}_{\bar{\pi}}^{\clubsuit}(s_{reset}, \phi(s_{reset})) = \gamma \cdot \mathcal{W}_{\tilde{d}_{\bar{\pi}}^{\clubsuit}}(\mathbf{P}_{\bar{\pi}}(\cdot | s), \bar{\mathbf{P}}_{\bar{\pi}}(\cdot | \phi(s))) = \gamma \cdot \tilde{d}_{\bar{\pi}}^{\clubsuit}(s_I, \bar{s}_I). \quad (3.3)$$

We obtain this equality by the fact that both $\mathbf{P}_{\bar{\pi}}(\cdot | s_{reset})$ and $\bar{\mathbf{P}}_{\bar{\pi}}(\cdot | \phi(s_{reset}))$ are Dirac measures at s_I and $\bar{s}_I$ respectively. Then, there is only one coupling in $\Pi(\mathbf{P}_{\bar{\pi}}(\cdot | s_{reset}), \bar{\mathbf{P}}_{\bar{\pi}}(\cdot | \phi(s_{reset})))$, which is the Dirac measure at $\langle s_I, \bar{s}_I \rangle$. By definition of the Wasserstein primal (Definition 33), we thus have $\mathcal{W}_{\tilde{d}_{\bar{\pi}}^{\clubsuit}}(\mathbf{P}_{\bar{\pi}}(\cdot | s), \bar{\mathbf{P}}_{\bar{\pi}}(\cdot | \phi(s))) = \tilde{d}_{\bar{\pi}}^{\clubsuit}(s_I, \bar{s}_I)$. Set $K_1 = 0, K_2 = 1$ if $\clubsuit = \ell$ and $K_1 = 1, K_2 = K_V$ otherwise, when $\clubsuit = \mathcal{R}$. Then by Lemma 13, we have

$$\begin{aligned} \tilde{d}_{\bar{\pi}}^{\clubsuit}(s_{reset}, \phi(s_{reset})) &\leq \frac{1}{\xi_{\bar{\pi}}(s_{reset})} \mathbb{E}_{s \sim \xi_{\bar{\pi}}} \tilde{d}_{\bar{\pi}}^{\clubsuit}(s, \phi(s)) \\ \iff \gamma \cdot \tilde{d}_{\bar{\pi}}^{\clubsuit}(s_I, \bar{s}_I) &\leq \frac{K_1 \cdot L_{\mathcal{R}} + \gamma K_2 \cdot L_{\mathbf{P}}}{\xi_{\bar{\pi}}(s_{reset})(1 - \gamma)}. \quad (\text{by Theorem 22 and Equation 3.3}) \end{aligned}$$

This yields the results for the bisimulation distances. Then, given any $\varphi \in \text{VLC}(\gamma)$, we have by Theorem 21 that $|V^{\bar{\pi}}(s_I, \varphi) - \bar{V}^{\bar{\pi}}(\bar{s}_I, \varphi)| \leq \tilde{d}_{\bar{\pi}}^{\ell}(s_I, \bar{s}_I)$.

It remains to show that the bound on the episodic return holds. To do this, we take a different approach. We intuitively use a proxy to the episodic value function, whose values only differ in the reset state. Define the value function $V_{\text{epis}^*}^{\bar{\pi}} \in \mathcal{V}$ as

$$V_{\text{epis}^*}^{\bar{\pi}}(s) = \mathcal{R}_{\bar{\pi}}(s) + \mathbb{E}_{s' \sim \mathbf{P}_{\bar{\pi}}(\cdot | s)} [V_{\text{epis}^*}^{\bar{\pi}}(s') \cdot \mathbb{1}\{s' \neq s_{reset}\}].$$

We define $\bar{V}_{\text{epis}\star}^{\bar{\pi}}$ similarly. By definition of the episodic return (Definition 18), it is clear that $V_{\text{epis}}^{\bar{\pi}}(s) = V_{\text{epis}\star}^{\bar{\pi}}(s)$ for all $s \neq s_{\text{reset}}$. Nevertheless, in s_{reset} , we have $V_{\text{epis}}^{\bar{\pi}}(s_{\text{reset}}) = 0 = V_{\text{epis}\star}^{\bar{\pi}}(s_{\text{reset}}) \cdot \mathbb{1}\{s \neq s_{\text{reset}}\}$. Then,

$$\begin{aligned}
& \mathbb{E}_{s \sim \xi_{\bar{\pi}}} \left| V_{\text{epis}\star}^{\bar{\pi}}(s) - \bar{V}_{\text{epis}\star}^{\bar{\pi}}(\phi(s)) \right| \\
& \leq \mathbb{E}_{s, \bar{a} \sim \xi_{\bar{\pi}}} \left| \mathcal{R}(s, \bar{a}) - \bar{\mathcal{R}}(\phi(s), \bar{a}) \right| + \\
& \quad \gamma \mathbb{E}_{s, \bar{a} \sim \xi_{\bar{\pi}}} \left| \mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, \bar{a})} \left[V_{\text{epis}\star}^{\bar{\pi}}(s') \cdot \mathbb{1}\{s' \neq s_{\text{reset}}\} \right] - \mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot | \phi(s), \bar{a})} \left[\bar{V}_{\text{epis}\star}^{\bar{\pi}}(\bar{s}') \cdot \mathbb{1}\{s \neq s_{\text{reset}}\} \right] \right| \\
& \hspace{15em} \text{(triangle inequality)} \\
& = \mathbb{E}_{s, \bar{a} \sim \xi_{\bar{\pi}}} \left| \mathcal{R}(s, \bar{a}) - \bar{\mathcal{R}}(\phi(s), \bar{a}) \right| + \gamma \mathbb{E}_{s, \bar{a} \sim \xi_{\bar{\pi}}} \left| \mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, \bar{a})} \left[V_{\text{epis}}^{\bar{\pi}}(s') \right] - \mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot | \phi(s), \bar{a})} \left[\bar{V}_{\text{epis}}^{\bar{\pi}}(\bar{s}') \right] \right| \\
& \leq L_{\mathcal{R}} + \gamma \mathbb{E}_{s \sim \xi_{\bar{\pi}}} \left| V_{\text{epis}}^{\bar{\pi}}(s) - \bar{V}_{\text{epis}}^{\bar{\pi}}(\phi(s)) \right| + \gamma K_V \cdot L_P \\
& \hspace{15em} \text{(by the same developments as in the proof of Lemma 11)} \\
& \leq L_{\mathcal{R}} + \gamma \frac{L_{\mathcal{R}} + \gamma K_V \cdot L_P}{1 - \gamma} + \gamma K_V \cdot L_P \hspace{10em} \text{(Lemma 11)} \\
& = L_{\mathcal{R}} \left(1 + \frac{\gamma}{1 - \gamma} \right) + \gamma K_V \cdot L_P \left(1 + \frac{\gamma}{1 - \gamma} \right) = \frac{L_{\mathcal{R}} + \gamma K_V \cdot L_P}{1 - \gamma}.
\end{aligned}$$

Finally, this result allows us to re-formalize the episodic value difference from the initial states through the values of the reset states:

$$\begin{aligned}
& \left| V_{\text{epis}}^{\bar{\pi}}(s_I) - \bar{V}_{\text{epis}}^{\bar{\pi}}(\bar{s}_I) \right| = \left| V_{\text{epis}\star}^{\bar{\pi}}(s_I) - \bar{V}_{\text{epis}\star}^{\bar{\pi}}(\bar{s}_I) \right| = \frac{1}{\gamma} \left| \gamma \cdot V_{\text{epis}\star}^{\bar{\pi}}(s_I) - \gamma \cdot \bar{V}_{\text{epis}\star}^{\bar{\pi}}(\bar{s}_I) \right| \\
& = \frac{1}{\gamma} \left| V_{\text{epis}\star}^{\bar{\pi}}(s_{\text{reset}}) - \bar{V}_{\text{epis}\star}^{\bar{\pi}}(\phi(s_{\text{reset}})) \right| \\
& \hspace{10em} \text{(by definition of } V_{\text{epis}\star}^{\bar{\pi}} \text{ and } \bar{V}_{\text{epis}\star}^{\bar{\pi}}, \text{ since } \mathbf{P}_{\bar{\pi}}(s_I | s_{\text{reset}}) = 1 = \bar{\mathbf{P}}_{\bar{\pi}}(\bar{s}_I | \phi(s_{\text{reset}}))) \\
& \leq \frac{1}{\gamma \xi_{\bar{\pi}}(s_{\text{reset}})} \mathbb{E}_{s \sim \xi_{\bar{\pi}}} \left| V_{\text{epis}\star}^{\bar{\pi}}(s) - \bar{V}_{\text{epis}\star}^{\bar{\pi}}(\phi(s)) \right| \hspace{10em} \text{(by Lemma 13)} \\
& \leq \frac{L_{\mathcal{R}}/\gamma + K_V \cdot L_P}{\xi_{\bar{\pi}}(s_{\text{reset}})(1 - \gamma)}. \hspace{10em} \text{(by the previous developments)}
\end{aligned}$$

This concludes the proof. $\square$

Representation Quality Guarantees

In addition to abstraction quality guarantees, we leverage the local losses to establish *representation quality guarantees*. Again, we formalize those guarantees as bisimulation and value difference bounds. We consider that the representation induced by the embedding function ϕ is well-suited for verification and RL when the latent space “preserves” the value functions. Precisely, states where the agent exhibits clearly different behaviors should be embedded to distinct discrete latent states. When this condition is not met, the most degenerate scenario occurs when every state is mapped to the same latent state, indicating that the latent space has collapsed to a single point. Conversely, states embedded to the same latent state should lead to close behaviors and thus have close values. This is precisely what happens when the local losses are minimized: states sharing the same abstract representation imply a close bisimulation distance, and thus close values.

Theorem 24: Representation quality

For any pair of measurable states $s_1, s_2 \in \mathcal{S}$ so that $\phi(s_1) = \phi(s_2)$,

$$\tilde{d}_{\pi}^{\ell}(s_1, s_2) \leq \frac{\gamma L_P}{1 - \gamma} \left(\frac{1}{\xi_{\pi}(s_1)} + \frac{1}{\xi_{\pi}(s_2)} \right), \quad \tilde{d}_{\pi}^{\mathcal{R}}(s_1, s_2) \leq \frac{L_R + \gamma K_V \cdot L_P}{1 - \gamma} \left(\frac{1}{\xi_{\pi}(s_1)} + \frac{1}{\xi_{\pi}(s_2)} \right).$$

(*) Furthermore, assuming $\mathcal{M}$ and $\overline{\mathcal{M}}$ are episodic processes,

$$\left| V_{\text{epis}}^{\pi}(s_1) - V_{\text{epis}}^{\pi}(s_2) \right| \leq \frac{L_R + \gamma K_V \cdot L_P}{1 - \gamma} \left(\frac{1}{\xi_{\pi}(s_1)} + \frac{1}{\xi_{\pi}(s_2)} \right).$$

Proof. Let $\clubsuit \in \{\ell, \mathcal{R}\}$ and $K_{\clubsuit} = K_V$ if $\clubsuit = \mathcal{R}$ and $K_{\clubsuit} = 1$ otherwise. Then, we have

$$\begin{aligned} & \tilde{d}_{\pi}^{\clubsuit}(s_1, s_2) \\ & \leq \tilde{d}_{\pi}^{\clubsuit}(s_1, \phi(s_1)) + \tilde{d}_{\pi}^{\clubsuit}(\phi(s_1), \phi(s_2)) + \tilde{d}_{\pi}^{\clubsuit}(\phi(s_2), s_2) \quad (\text{triangle inequality}) \\ & \leq \xi_{\pi}(s_1)^{-1} \mathbb{E}_{s \sim \xi_{\pi}} \tilde{d}_{\pi}^{\clubsuit}(s, \phi(s)) + \xi_{\pi}(s_2)^{-1} \mathbb{E}_{s \sim \xi_{\pi}} \tilde{d}_{\pi}^{\clubsuit}(s, \phi(s)) + \tilde{d}_{\pi}^{\clubsuit}(\phi(s_1), \phi(s_2)) \quad (\text{Lemma 13}) \\ & \leq (\xi_{\pi}(s_1)^{-1} + \xi_{\pi}(s_2)^{-1}) \left(\frac{\mathbb{1}\{\clubsuit = \mathcal{R}\} \cdot L_R + \gamma K_{\clubsuit} \cdot L_P}{1 - \gamma} \right) + \tilde{d}_{\pi}^{\clubsuit}(\phi(s_1), \phi(s_2)) \quad (\text{Theorem 22}) \\ & \leq (\xi_{\pi}(s_1)^{-1} + \xi_{\pi}(s_2)^{-1}) \left(\frac{\mathbb{1}\{\clubsuit = \mathcal{R}\} \cdot L_R + \gamma K_{\clubsuit} \cdot L_P}{1 - \gamma} \right). \quad (\phi(s_1) = \phi(s_2) \text{ by assumption}) \end{aligned}$$

The proof for the episodic value difference can be obtained by mimicking this proof and replacing every occurrence of $\tilde{d}_{\pi}^{\clubsuit}$ with the episodic value difference. $\square$

As the values are Lipschitz continuous with respect to the bisimulation distance (Equation 3.1), we consequently retrieve value difference bounds, akin the those established by Gelada et al. [2019] for the expected return in the context of Lipschitz continuous latent space models (DeepMDPs, cf. Remark 17).

Corollary 3: Representation quality as value difference bounds

For any pair of measurable states $s_1, s_2 \in S$ so that $\phi(s_1) = \phi(s_2)$, we have for any $\varphi \in \text{VLC}(\gamma)$ that

$$|V^{\bar{\pi}}(s_1, \varphi) - V^{\bar{\pi}}(s_2, \varphi)| \leq \frac{\gamma L_P}{1 - \gamma} \left(\frac{1}{\xi_{\bar{\pi}}(s_1)} + \frac{1}{\xi_{\bar{\pi}}(s_2)} \right), \text{ and}$$

$$|V^{\bar{\pi}}(s_1) - V^{\bar{\pi}}(s_2)| \leq \frac{L_R + \gamma K_V \cdot L_P}{1 - \gamma} \left(\frac{1}{\xi_{\bar{\pi}}(s_1)} + \frac{1}{\xi_{\bar{\pi}}(s_2)} \right).$$

Remark 18: Interpreting the bounds

When $\mathcal{M}_{\bar{\pi}}$ is irreducible and recurrent (e.g., episodic), then $1/\xi_{\bar{\pi}}(s_1)$ and $1/\xi_{\bar{\pi}}(s_2)$ respectively represent the *mean recurrence time* to s_1 and s_2 under $\bar{\pi}$, i.e., the expected number of time steps to return to s_1 and s_2 when $\bar{\pi}$ is executed (Equation 3.2).

1.3 Quantifying the Quality of the Abstraction

The previous section establishes bounds on the quality of the abstraction based on L_R, L_P , and $\xi_{\bar{\pi}}(s_{\text{reset}})$. Computing these quantities or probably approximately correctly (PAC) estimating them, however, is intractable in practice. Notably, the expectation of the Wasserstein term in $L_P = \mathbb{E}_{s, \bar{a} \sim \xi_{\bar{\pi}}} \mathcal{W}_{\bar{a}}(\phi P(\cdot | s, \bar{a}), \bar{P}(\cdot | \phi(s), \bar{a}))$ and the fact that the transition probabilities P of the original environment are unknown deem the computation infeasible. Intuitively, while estimating Wasserstein and total variation distances from samples is feasible [Genevay, 2019], achieving this necessitates access to a generative model for $P(\cdot | s, a)$ [Kearns et al., 2002] — which is not compliant to our setting. This entails drawing a significant number of samples s' from $P(\cdot | s, a)$ for each state-action pair s, a drawn from $\xi_{\bar{\pi}}$. These samples are essential for initially estimating the inner Wasserstein term, from which the overall outer expectation needs then to be estimated. Notably, the errors may accumulate during the estimation process, particularly in the inner calculation, making the overall estimation error of L_P difficult to control.

While those issues can be avoided by assuming that the MDP is deterministic, i.e., $P(\cdot | s, a)$ is a Dirac measure for all $s \in S, a \in \mathcal{A}$ (as done by Gelada et al. [2019] in their practical setting), this assumption is too restrictive in general, for many practical

applications. To deal with general MDPs, we study an upper bound on L_P that can be efficiently approximated from samples:

$$L_P \leq \mathbb{E}_{s, \bar{a}, s' \sim \xi_{\bar{\pi}}} \mathcal{W}_{\bar{a}}(\phi(\cdot | s'), \bar{\mathbb{P}}(\cdot | \phi(s), \bar{a})) = L_P^\uparrow, \quad (3.4)$$

where $\phi(\cdot | s)$ is merely defined as the Dirac measure at $\phi(s)$, i.e., $\phi(\bar{s} | s) = \mathbb{1}\{\bar{s} = \phi(s)\}$, for any $s \in \mathcal{S}$ and $\bar{s} \in \bar{\mathcal{S}}$. The inequality in Equation 3.4 is a consequence of Lemma 10:

$$\mathbb{E}_{s, \bar{a} \sim \xi_{\bar{\pi}}} \left[\sup_f \mathbb{E}_{s' \sim \mathbb{P}(\cdot | s, \bar{a})} f(\phi(s')) - \mathbb{E}_{\bar{s}' \sim \bar{\mathbb{P}}(\cdot | \phi(s), \bar{a})} f(\bar{s}') \right] \leq \mathbb{E}_{s, \bar{a}, s' \sim \xi_{\bar{\pi}}} \left[\sup_f f(\phi(s')) - \mathbb{E}_{\bar{s}' \sim \bar{\mathbb{P}}(\cdot | \phi(s), \bar{a})} f(\bar{s}') \right].$$

We further obtain PAC bounds on L_P , L_R , and $\xi_{\bar{\pi}}(s_{reset})$ by simulating $\mathcal{M}$. Intuitively, we simulate the MDP to draw transitions from the stationary distribution $\xi_{\bar{\pi}}$. Using classical randomized algorithms, one can efficiently sample from the stationary distribution [Lovász and Winkler, 1995; Propp and Wilson, 1998] even for unknown probabilities (cf. Chapter 2, Section 2.1 for a discussion in the episodic context). We then use the empirical distribution of the transitions to estimate L_R and L_P . When considering episodic processes, the estimate for $\xi_{\bar{\pi}}(s_{reset})$ is similarly obtained by taking the portion of visits to s_{reset} in a simulation. The estimate for L_P is formalized as an upper bound which follows from $L_P^\uparrow$. Intuitively, every time the simulation proceeds from s to s' following an action a , we derive a “reward” of $\bar{\mathbb{P}}(\phi(s') | \phi(s), a)$. Since L_P is a loss, we obtain the estimate by subtracting the average reward from 1.

Lemma 14: PAC bounds for the local losses

Let $\{\langle s_t, \bar{a}_t, r_t, s'_t \rangle : 1 \leq t \leq T\}$ be a collection of T transitions drawn from $\xi_{\bar{\pi}}$ by simulating $\mathcal{M}_{\bar{\pi}}$, and $\varepsilon, \delta > 0$. Define $K_R = \max\{4R_{\text{MAX}}^2, 1\}$,

$$\hat{L}_R = \frac{1}{T} \sum_{t=1}^T |r_t - \bar{R}(\phi(s_t), \bar{a}_t)|, \text{ and } \hat{L}_P = 1 - \frac{1}{T} \sum_{t=1}^T \bar{\mathbb{P}}(\phi(s'_t) | \phi(s_t), \bar{a}_t).$$

With at least probability $1 - \delta$, we have

- (i) if $T \geq \lceil -\log \delta / 2\varepsilon^2 \rceil$, then $\hat{L}_P + \varepsilon > L_P$; and
- (ii) if $T \geq \lceil -K_R \log(\delta/2) / 2\varepsilon^2 \rceil$, then $\hat{L}_R + \varepsilon > L_R$ and $\hat{L}_P + \varepsilon > L_P$.

Lemma 14: PAC bounds for the local losses (continued)

(*) Furthermore, assume $\mathcal{M}$ and $\overline{\mathcal{M}}$ are episodic. Define

$$\hat{\xi}_{reset} = 1/T \cdot \sum_{t=0}^T \mathbb{1}\{s_t = s_{reset}\};$$

with at least probability $1 - \delta$, we have

(iii) if $T \geq \lceil -\log(\delta/2)/2\varepsilon^2 \rceil$, then $\hat{L}_P + \varepsilon > L_P$ and $\xi_{\bar{\pi}}(s_{reset}) > \hat{\xi}_{reset} - \varepsilon$; and

(iv) if $T \geq \lceil -K_R \log(\delta/3)/2\varepsilon^2 \rceil$, then $\hat{L}_R + \varepsilon > L_R$, $\hat{L}_P + \varepsilon > L_P$, and $\xi_{\bar{\pi}}(s_{reset}) > \hat{\xi}_{reset} - \varepsilon$.

Proof. We start by recalling the Hoeffding's inequality.

Let $X_1, \dots, X_n$ be independent random variables such that $a \leq X_i \leq b$ for all $i \in \{1, \dots, n\}$ and $a, b \in \mathbb{R}$. Then, for any $\varepsilon > 0$,

$$\begin{aligned} \frac{1}{n} \sum_{i=1}^n X_i - \mathbb{E}[X_i] &\geq \varepsilon && \text{with at most probability } \exp\left(\frac{-2n\varepsilon^2}{(b-a)^2}\right), \text{ and} \\ \frac{1}{n} \sum_{i=1}^n X_i - \mathbb{E}[X_i] &\leq -\varepsilon && \text{with at most probability } \exp\left(\frac{-2n\varepsilon^2}{(b-a)^2}\right). \end{aligned} \quad (3.5)$$

Recall that the Wasserstein distance coincides with total variation when the state space is discrete. Then by definition of the total variation distance, we have

$$\begin{aligned} L_P &= \mathbb{E}_{s, a \sim \xi_{\bar{\pi}}} \mathcal{W}_{\bar{d}}(\phi \mathbf{P}(\cdot | s, a), \bar{\mathbf{P}}(\cdot | \phi(s), a)) \\ &= \mathbb{E}_{s, a \sim \xi_{\bar{\pi}}} \left[\frac{1}{2} \sum_{\bar{s}' \in \bar{\mathcal{S}}} \left| \mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, a)} \mathbb{1}\{\phi(s') = \bar{s}'\} - \bar{\mathbf{P}}(\bar{s}' | s, a) \right| \right]. \end{aligned}$$

Notice that this quantity cannot be approximated from samples distributed according to $\xi_{\bar{\pi}}$ alone: intuitively, we need to have access to the original transition function $\mathbf{P}$ to be able to estimate the expectation $\mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, a)} \mathbb{1}\{\phi(s') = \bar{s}'\}$ for every single sample of $\xi_{\bar{\pi}}$.

Instead, consider the upper bound on L_P :

$$\begin{aligned} L_P &\leq \mathbb{E}_{s, a \sim \xi_{\bar{\pi}}} \mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, a)} \left[\frac{1}{2} \sum_{\bar{s}' \in \bar{\mathcal{S}}} \left| \mathbb{1}\{\phi(s') = \bar{s}'\} - \bar{\mathbf{P}}(\bar{s}' | s, a) \right| \right] && \text{(Jensen's inequality)} \\ &= \mathbb{E}_{s, a \sim \xi_{\bar{\pi}}} \mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, a)} \mathcal{W}_{\bar{d}}(\phi(\cdot | s'), \bar{\mathbf{P}}(\cdot | \bar{s}, a)) = L_P^{\uparrow}, \end{aligned}$$

where $\phi(\bar{s}' \mid s')$ is defined as $\mathbb{1}\{\phi(s') = \bar{s}'\}$ for any $\bar{s}' \in \bar{\mathcal{S}}$. Then,

$$\begin{aligned}
 L_P^\dagger &= \mathbb{E}_{s, \bar{a}, s' \sim \xi_\pi} \mathcal{W}_{\bar{a}}(\phi(\cdot \mid s'), \bar{\mathbf{P}}(\cdot \mid \phi(s), a)) \\
 &= \mathbb{E}_{s, \bar{a}, s' \sim \xi_\pi} \left[\frac{1}{2} \sum_{\bar{s}' \in \bar{\mathcal{S}}} \left| \phi(\bar{s}' \mid s') - \bar{\mathbf{P}}(\bar{s}' \mid \phi(s), \bar{a}) \right| \right] \\
 &= \mathbb{E}_{s, \bar{a}, s' \sim \xi_\pi} \left[\frac{1}{2} \cdot \left((1 - \bar{\mathbf{P}}(\phi(s') \mid \phi(s), \bar{a})) + \sum_{\bar{s}' \in \bar{\mathcal{S}} \setminus \{\phi(s')\}} \left| 0 - \bar{\mathbf{P}}(\bar{s}' \mid \phi(s), \bar{a}) \right| \right) \right] \\
 &\quad \text{(because } \phi(\bar{s}' \mid s') = 1 \text{ if } \phi(s') = \bar{s}' \text{ and 0 otherwise)} \\
 &= \mathbb{E}_{s, \bar{a}, s' \sim \xi_\pi} \left[\frac{1}{2} \cdot \left((1 - \bar{\mathbf{P}}(\phi(s') \mid \phi(s), \bar{a})) + \sum_{\bar{s}' \in \bar{\mathcal{S}} \setminus \{\phi(s')\}} \bar{\mathbf{P}}(\bar{s}' \mid \phi(s), \bar{a}) \right) \right] \\
 &= \mathbb{E}_{s, \bar{a}, s' \sim \xi_\pi} \left[\frac{1}{2} \cdot 2 \cdot (1 - \bar{\mathbf{P}}(\phi(s') \mid \phi(s), \bar{a})) \right] \\
 &= \mathbb{E}_{s, \bar{a}, s' \sim \xi_\pi} [1 - \bar{\mathbf{P}}(\phi(s') \mid \phi(s), \bar{a})].
 \end{aligned}$$

By the Hoeffding's inequality (Equation 3.5), we obtain that $\hat{L}_P + \varepsilon \leq L_P^\dagger$ with probability at most $\exp(-2T\varepsilon^2)$. Equivalently, this means that $\hat{L}_P + \varepsilon > L_P^\dagger$ with at least probability $1 - \exp(-2T\varepsilon^2)$. The fact that $\hat{L}_P + \varepsilon > L_P^\dagger \geq L_P$ yields the bound on L_P .

For the rewards, we obtain the bound via a plain application of the Hoeffding's inequality. Note that the rewards are comprised in the interval $[-R_{\text{MAX}}, R_{\text{MAX}}]$, thus $|r_t - \bar{\mathcal{R}}(\phi(s_t), \bar{a}_t)| \in [0, 2R_{\text{MAX}}]$. Then, we have with at most probability $\exp(-2T\varepsilon^2/4R_{\text{MAX}}^2)$ that $\hat{L}_R + \varepsilon \leq L_R$. Therefore, by the union bound we have

$$\Pr(\hat{L}_P + \varepsilon \leq L_P^\dagger \text{ or } \hat{L}_R + \varepsilon \leq L_R) \leq \exp(-2T\varepsilon^2) + \exp(-2T\varepsilon^2/4R_{\text{MAX}}^2).$$

Recall that our purpose is to allow an error beyond ε with at most probability δ . Then, we need to find an upper bound on T so that

$$\delta \geq \exp(-2T\varepsilon^2) + \exp(-2T\varepsilon^2/4R_{\text{MAX}}^2). \quad (3.6)$$

We consider two cases. First, assume $R_{\text{MAX}} \leq 1/2$, then $\exp(-2T\varepsilon^2/4R_{\text{MAX}}^2) < \exp(-2T\varepsilon^2)$ for any $T \geq 0$. Then, finding a $T \geq 0$ which yields $\delta \geq 2\exp(-2T\varepsilon^2)$ is sufficient to ensure the bound of Equation 3.6. In that case, we have

$$\delta \geq 2\exp(-2T\varepsilon^2) \Leftrightarrow \delta/2 \geq \exp(-2T\varepsilon^2) \Leftrightarrow \log(\delta/2) \geq -2T\varepsilon^2 \Leftrightarrow T \geq \frac{-\log(\delta/2)}{2\varepsilon^2}. \quad (3.7)$$

Second, assume $R_{\text{MAX}} > 1/2$, then we have that $\exp(-2T\varepsilon^2) < \exp(-2T\varepsilon^2/4R_{\text{MAX}}^2)$ for any $T \geq 0$. Therefore, finding a $T \geq 0$ which yields $\delta \geq 2\exp(-2T\varepsilon^2/4R_{\text{MAX}}^2)$ is sufficient to ensure the bound of Equation 3.6. Then, we have

$$\begin{aligned} \delta \geq 2\exp(-2T\varepsilon^2/4R_{\text{MAX}}^2) &\iff \delta/2 \geq \exp(-2T\varepsilon^2/4R_{\text{MAX}}^2) \iff \log(\delta/2) \geq \frac{-2T\varepsilon^2}{4R_{\text{MAX}}^2} \\ \iff T &\geq \frac{-2R_{\text{MAX}}^2 \log(\delta/2)}{\varepsilon^2}. \end{aligned}$$

Therefore, setting $T \geq \lceil -K_R \log(\delta/2)/2\varepsilon^2 \rceil$ is sufficient to ensure

$$\begin{aligned} &\Pr(\hat{L}_P + \varepsilon \leq L_P^\uparrow \text{ or } \hat{L}_R + \varepsilon \leq L_R) \leq \delta \\ \iff &\Pr(\hat{L}_P + \varepsilon > L_P^\uparrow \text{ and } \hat{L}_R + \varepsilon > L_R) \geq 1 - \delta \\ \iff &\Pr(\hat{L}_P + \varepsilon > L_P \text{ and } \hat{L}_R + \varepsilon > L_R) \geq 1 - \delta. \end{aligned}$$

Assume now $\mathcal{M}$ and $\overline{\mathcal{M}}$ are episodic. Again, by applying the Hoeffding's inequality, we obtain that with at most probability $\exp(-2T\varepsilon^2)$, we have $\hat{\xi}_{\text{reset}} - \varepsilon \geq \xi_{\overline{\pi}}(s_{\text{reset}})$. By the union bound, we have

$$\Pr(\hat{L}_P + \varepsilon \leq L_P^\uparrow \text{ or } \hat{\xi}_{\text{reset}} - \varepsilon \geq \xi_{\overline{\pi}}(s_{\text{reset}})) \leq \exp(-2T\varepsilon^2) + \exp(-2T\varepsilon^2).$$

Then, by the same reasoning as above (Equation 3.7), we have that with at least probability $1 - \delta$, $\hat{L}_P + \varepsilon > L_P$ and $\hat{\xi}_{\text{reset}} - \varepsilon < \xi_{\overline{\pi}}(s_{\text{reset}})$ if $T \geq \lceil -\log(\delta/2)/2\varepsilon^2 \rceil$. Finally, let us consider the case where L_R is also estimated. By the union bound, we have

$$\begin{aligned} &\Pr(\hat{L}_P + \varepsilon \leq L_P^\uparrow \text{ or } \hat{L}_R + \varepsilon \leq L_R \text{ or } \hat{\xi}_{\text{reset}} - \varepsilon \geq \xi_{\overline{\pi}}(s_{\text{reset}})) \\ &\leq \exp(-2T\varepsilon^2) + \exp(-2T\varepsilon^2/4R_{\text{MAX}}^2) + \exp(-2T\varepsilon^2). \end{aligned}$$

We again differentiate two cases. By the same reasoning as above, if $R_{\text{MAX}} \leq 1/2$, then it sufficient to ensure that $\delta \geq 3\exp(-2T\varepsilon^2)$, in which case we need to set $T \geq \lceil -\log(\delta/3)/2\varepsilon^2 \rceil$. If $R_{\text{MAX}} > 1/2$, then it sufficient to ensure that $\delta \geq 3\exp(-2T\varepsilon^2/4R_{\text{MAX}}^2)$, in which case we need to set $T \geq \lceil -2R_{\text{MAX}}^2 \log(\delta/3)/\varepsilon^2 \rceil$. Then, we take $T \geq \lceil -K_R \log(\delta/3)/2\varepsilon^2 \rceil$. $\square$

Now that we can PAC estimate the local losses, we can further use these estimates to obtain PAC bounds on the abstraction quality. The following theorem establishes a sufficient number of simulation steps to provide a PAC bound on the abstraction quality in terms of the bisimulation distances and value functions.

Theorem 25: PAC abstraction quality bounds (bis)

Let $\{\langle s_t, \bar{a}_t, r_t, s'_t \rangle : 1 \leq t \leq T\}$ be a collection of T transitions drawn from $\xi_{\bar{\pi}}$ by simulating $\mathcal{M}_{\bar{\pi}}$, $\varepsilon, \delta > 0$, and $\varphi \in \text{VLC}(\gamma)$. Define K_R , $\hat{L}_P$, $\hat{L}_R$, and $\hat{\xi}_{reset}$ as in Lemma 14. Then, with at least probability $1 - \delta$, we have

$$(i) \text{ if } T \geq \left\lceil \frac{-\gamma^2 \log \delta}{2\varepsilon^2(1-\gamma)^2} \right\rceil,$$

$$\mathbb{E}_{s \sim \xi_{\bar{\pi}}} |V^{\bar{\pi}}(s, \varphi) - \bar{V}^{\bar{\pi}}(\phi(s), \varphi)| \leq \mathbb{E}_{s \sim \xi_{\bar{\pi}}} \tilde{d}_{\bar{\pi}}^{\ell}(s, \phi(s)) \leq \frac{\gamma \hat{L}_P}{1-\gamma} + \varepsilon;$$

$$(ii) \text{ if } T \geq \left\lceil \frac{-K_R(1+\gamma K_V)^2 \log(\delta/2)}{2\varepsilon^2(1-\gamma)^2} \right\rceil,$$

$$\mathbb{E}_{s \sim \xi_{\bar{\pi}}} |V^{\bar{\pi}}(s) - \bar{V}^{\bar{\pi}}(\phi(s))| \leq \mathbb{E}_{s \sim \xi_{\bar{\pi}}} \tilde{d}_{\bar{\pi}}^R(s, \phi(s)) \leq \frac{\hat{L}_R + \gamma K_V \cdot \hat{L}_P}{1-\gamma} + \varepsilon.$$

(*) Furthermore, assuming $\mathcal{M}$ and $\bar{\mathcal{M}}$ are episodic,

$$(iii) \text{ if } T \geq \left\lceil \frac{-\log(\delta/2) \left(\hat{L}_P + \hat{\xi}_{reset}(1+\varepsilon(1-\gamma)) \right)^2}{2\hat{\xi}_{reset}^4 \varepsilon^2 (1-\gamma)^2} \right\rceil,$$

$$|V^{\bar{\pi}}(s_I, \varphi) - \bar{V}^{\bar{\pi}}(\bar{s}_I, \varphi)| \leq \tilde{d}_{\bar{\pi}}^{\ell}(s_I, \bar{s}_I) \leq \frac{\hat{L}_P}{\hat{\xi}_{reset}(1-\gamma)} + \varepsilon;$$

$$(iv) \text{ if } T \geq \left\lceil \frac{-K_R \log(\delta/3) \left(\hat{L}_R + \gamma K_V \hat{L}_P + \hat{\xi}_{reset}(1+\gamma K_V + \gamma \varepsilon(1-\gamma)) \right)^2}{2\gamma^2 \hat{\xi}_{reset}^4 \varepsilon^2 (1-\gamma)^2} \right\rceil,$$

$$\tilde{d}_{\bar{\pi}}^R(s_I, \bar{s}_I) \leq \frac{\hat{L}_R/\gamma + K_V \hat{L}_P}{\hat{\xi}_{reset}(1-\gamma)} + \varepsilon, \text{ and } |V_{\text{epis}}^{\bar{\pi}}(s_I) - \bar{V}_{\text{epis}}^{\bar{\pi}}(\bar{s}_I)| \leq \frac{\hat{L}_R/\gamma + K_V \hat{L}_P}{\hat{\xi}_{reset}(1-\gamma)} + \varepsilon.$$

Proof. Let $\varepsilon' \in (0, 1)$ be another error parameter. We know by Theorem 22 and Lemma 14 that when a sufficient number of samples T is collected, we have

(i) $\mathbb{E}_{s \sim \xi_{\bar{\pi}}} \tilde{d}_{\bar{\pi}}^{\ell}(s, \phi(s)) \leq \frac{\gamma \hat{L}_P}{1-\gamma} \leq \frac{\gamma(\hat{L}_P + \varepsilon')}{1-\gamma}$ with at least probability $1 - \delta$. Then, to ensure an error of at most ε , we need to set ε' such that:

$$\frac{\gamma(\hat{L}_P + \varepsilon')}{1-\gamma} \leq \frac{\gamma \hat{L}_P}{1-\gamma} + \varepsilon \iff \frac{\gamma \varepsilon'}{1-\gamma} \leq \varepsilon \iff \varepsilon' \leq \frac{\varepsilon(1-\gamma)}{\gamma}.$$

By Lemma 14, we need $T \geq \left\lceil \frac{-\log \delta}{2\varepsilon'^2} \right\rceil \geq \left\lceil \frac{-\gamma^2 \log \delta}{2\varepsilon^2(1-\gamma)^2} \right\rceil$ samples to provide an error of at most ε with at least probability $1 - \delta$.

(ii) $\mathbb{E}_{s \sim \xi_{\pi}} \tilde{d}_{\pi}^R(s, \phi(s)) \leq \frac{L_R + \gamma K_V \cdot L_P}{1 - \gamma} \leq \frac{(\hat{L}_R + \varepsilon') + \gamma K_V \cdot (\hat{L}_P + \varepsilon')}{1 - \gamma} = \frac{\hat{L}_R + \gamma K_V \cdot \hat{L}_P + \varepsilon' (1 + \gamma K_V)}{1 - \gamma}$ with at least probability $1 - \delta$. Then, to ensure an error of at most ε , we need to set ε' such that:

$$\frac{\varepsilon' + \gamma K_V \varepsilon'}{1 - \gamma} \leq \varepsilon \Leftrightarrow \varepsilon' + \gamma K_V \varepsilon' \leq \varepsilon(1 - \gamma) \Leftrightarrow \varepsilon'(1 + \gamma K_V) \leq \varepsilon(1 - \gamma) \Leftrightarrow \varepsilon' \leq \frac{\varepsilon(1 - \gamma)}{1 + \gamma K_V}.$$

By Lemma 14, we need $T \geq \lceil -K_R \log(\delta/2) / 2\varepsilon'^2 \rceil \geq \left\lceil \frac{-K_R(1 + \gamma K_V)^2 \log(\delta/2)}{2\varepsilon^2(1 - \gamma)^2} \right\rceil$ samples to ensure that with at least probability $1 - \delta$ to ensure an error of at most ε .

Furthermore, by Theorem 23, we have

(iii) $\tilde{d}_{\pi}^{\ell}(s_I, \bar{s}_I) \leq \frac{L_P}{\hat{\xi}_{\pi}(s_{\text{reset}})(1 - \gamma)} \leq \frac{\hat{L}_P + \varepsilon'}{(\hat{\xi}_{\text{reset}} - \varepsilon')(1 - \gamma)}$ with probability at least $1 - \delta$. Then, to ensure an error of at most $\varepsilon > 0$, we need to set ε' such that:

$$\begin{aligned} \frac{\hat{L}_P}{\hat{\xi}_{\text{reset}} \cdot (1 - \gamma)} + \varepsilon &\geq \frac{\hat{L}_P + \varepsilon'}{(\hat{\xi}_{\text{reset}} - \varepsilon')(1 - \gamma)} \\ \Leftrightarrow (\hat{\xi}_{\text{reset}} - \varepsilon') \left(\frac{\hat{L}_P}{\hat{\xi}_{\text{reset}}} + \varepsilon(1 - \gamma) \right) &\geq \hat{L}_P + \varepsilon' \\ \Leftrightarrow \hat{L}_P + \hat{\xi}_{\text{reset}} \cdot \varepsilon(1 - \gamma) - \frac{\hat{L}_P \cdot \varepsilon'}{\hat{\xi}_{\text{reset}}} - \varepsilon \cdot \varepsilon'(1 - \gamma) &\geq \hat{L}_P + \varepsilon' \\ \Leftrightarrow \hat{\xi}_{\text{reset}} \cdot \varepsilon(1 - \gamma) \geq \varepsilon' + \frac{\hat{L}_P \cdot \varepsilon'}{\hat{\xi}_{\text{reset}}} + \varepsilon \cdot \varepsilon'(1 - \gamma) &= \varepsilon' \left(1 + \frac{\hat{L}_P}{\hat{\xi}_{\text{reset}}} + \varepsilon(1 - \gamma) \right) \\ \Leftrightarrow \frac{\hat{\xi}_{\text{reset}} \cdot \varepsilon(1 - \gamma)}{1 + \frac{\hat{L}_P}{\hat{\xi}_{\text{reset}}} + \varepsilon(1 - \gamma)} \geq \varepsilon' &\Leftrightarrow \frac{\hat{\xi}_{\text{reset}}^2 \cdot \varepsilon(1 - \gamma)}{\hat{L}_P + \hat{\xi}_{\text{reset}} \cdot (1 + \varepsilon(1 - \gamma))} \geq \varepsilon'. \end{aligned}$$

Notice that this upper bound on $\varepsilon' > 0$ is well defined since

$$(a) \hat{\xi}_{\text{reset}}^2 \cdot \varepsilon(1 - \gamma) > 0, \text{ and} \quad (b) \hat{L}_P + \hat{\xi}_{\text{reset}}(1 + \varepsilon(1 - \gamma)) > 0.$$

Then, setting $\varepsilon' \leq \frac{\hat{\xi}_{\text{reset}}^2 \cdot \varepsilon(1 - \gamma)}{\hat{L}_P + \hat{\xi}_{\text{reset}}(1 + \varepsilon(1 - \gamma))}$ means by Lemma 14 that we need

$$T \geq \left\lceil \frac{-\log(\delta/2)}{2\varepsilon'^2} \right\rceil \geq \left\lceil \frac{-\log(\delta/2) \left(\hat{L}_P + \hat{\xi}_{\text{reset}}(1 + \varepsilon(1 - \gamma)) \right)^2}{2\hat{\xi}_{\text{reset}}^4 \cdot \varepsilon^2(1 - \gamma)^2} \right\rceil$$

samples to provide an error of at most ε with probability at least $1 - \delta$.

(iv) $\tilde{d}_{\pi}^R(s_I, \bar{s}_I) \leq \frac{\hat{L}_R/\gamma + K_V \cdot \hat{L}_P}{\hat{\xi}_{reset}(1-\gamma)} \leq \frac{\hat{L}_R + \varepsilon'/\gamma + K_V \cdot (\hat{L}_P + \varepsilon')}{(\hat{\xi}_{reset} - \varepsilon')(1-\gamma)}$ with probability at least $1 - \delta$. Then, to

ensure an error of at most ε , we need to set ε' such that:

$$\begin{aligned}
 & \frac{\hat{L}_R/\gamma + K_V \cdot \hat{L}_P}{\hat{\xi}_{reset} \cdot (1-\gamma)} + \varepsilon \geq \frac{\hat{L}_R + \varepsilon'/\gamma + K_V \cdot (\hat{L}_P + \varepsilon')}{(\hat{\xi}_{reset} - \varepsilon')(1-\gamma)} \\
 \Leftrightarrow & \frac{\hat{L}_R + \gamma K_V \cdot \hat{L}_P}{\hat{\xi}_{reset} \cdot (1-\gamma)} + \gamma \varepsilon \geq \frac{\hat{L}_R + \varepsilon' + \gamma K_V \cdot (\hat{L}_P + \varepsilon')}{(\hat{\xi}_{reset} - \varepsilon')(1-\gamma)} \\
 \Leftrightarrow & (\hat{\xi}_{reset} - \varepsilon') \left(\frac{\hat{L}_R + \gamma K_V \cdot \hat{L}_P}{\hat{\xi}_{reset}} + \gamma \varepsilon (1-\gamma) \right) \geq \hat{L}_R + \varepsilon' + \gamma K_V \cdot \hat{L}_P + \gamma K_V \varepsilon' \\
 \Leftrightarrow & \hat{L}_R + \gamma K_V \cdot \hat{L}_P + \hat{\xi}_{reset} \cdot \gamma \varepsilon (1-\gamma) - \frac{\varepsilon' \cdot (\hat{L}_R + \gamma K_V \cdot \hat{L}_P)}{\hat{\xi}_{reset}} - \gamma \varepsilon \cdot \varepsilon' (1-\gamma) \\
 & \geq \hat{L}_R + \gamma K_V \cdot \hat{L}_P + \varepsilon' (1 + \gamma K_V) \\
 \Leftrightarrow & \hat{\xi}_{reset} \cdot \gamma \varepsilon (1-\gamma) \geq \frac{\varepsilon' \cdot (\hat{L}_R + \gamma K_V \cdot \hat{L}_P)}{\hat{\xi}_{reset}} + \gamma \varepsilon \cdot \varepsilon' (1-\gamma) + \varepsilon' (1 + \gamma K_V) \\
 & = \varepsilon' \left(1 + \gamma K_V + \gamma \varepsilon (1-\gamma) + \frac{\hat{L}_R + \gamma K_V \hat{L}_P}{\hat{\xi}_{reset}} \right) \\
 \Leftrightarrow & \hat{\xi}_{reset}^2 \cdot \gamma \varepsilon (1-\gamma) \geq \varepsilon' \left(\hat{\xi}_{reset} \cdot (1 + \gamma K_V + \gamma \varepsilon (1-\gamma)) + \hat{L}_R + \gamma K_V \hat{L}_P \right) \\
 \Leftrightarrow & \frac{\gamma \hat{\xi}_{reset}^2 \cdot \varepsilon (1-\gamma)}{\hat{\xi}_{reset} \cdot (1 + \gamma K_V + \gamma \varepsilon (1-\gamma)) + \hat{L}_R + \gamma K_V \hat{L}_P} \geq \varepsilon'.
 \end{aligned}$$

Again, notice that this upper bound on $\varepsilon' > 0$ is well defined since

(a) $\gamma \hat{\xi}_{reset}^2 \cdot \varepsilon (1-\gamma) > 0$, and

(b) $\hat{\xi}_{reset} \cdot (1 + \gamma K_V + \gamma \varepsilon (1-\gamma)) + \hat{L}_R + \gamma K_V \hat{L}_P > 0$.

By Lemma 14, setting $\varepsilon' \leq \frac{\gamma \hat{\xi}_{reset}^2 \cdot \varepsilon (1-\gamma)}{\hat{\xi}_{reset} \cdot (1 + \gamma K_V + \gamma \varepsilon (1-\gamma)) + \hat{L}_R + \gamma K_V \hat{L}_P}$ means that we need

$$T \geq \left\lceil \frac{-K_R \cdot \log(\delta/3)}{2\varepsilon'^2} \right\rceil \geq \left\lceil \frac{-K_R \cdot \log(\delta/3) \left(\hat{L}_R + \gamma K_V \hat{L}_P + \hat{\xi}_{reset} \cdot (1 + \gamma K_V + \gamma \varepsilon (1-\gamma)) \right)^2}{2\gamma^2 \hat{\xi}_{reset}^4 \cdot \varepsilon^2 (1-\gamma)^2} \right\rceil$$

samples to provide an error of at most ε with probability at least $1 - \delta$. The bound for the episodic return is identical and can be obtained via the same reasoning. $\square$

Remark 19: Step bound as termination criterion

It is worth noting that the bound T acts as a termination criterion in any algorithm that estimates the bounds by simulating $\mathcal{M}_{\bar{\pi}}$. Notably, in the episodic cases (iii and iv), T *does rely on the current approximations of $L_{\mathcal{R}}$, $L_{\mathcal{P}}$, and $\xi_{\bar{\pi}}$* . Therefore, in contrast to the *average* bisimulation bounds (i and ii), where a sufficient number of samples T can be pre-computed from ε and δ , the bisimulation bounds *from the initial states* can be approximated using a probabilistic algorithm that terminates almost surely once T is exceeded, but without a predetermined endpoint.

Discussion. The PAC bounds presented in this section are useful to quantify a notion of confidence in the quality of the abstraction. When a latent space model and a latent policy are provided, one can simulate the original environment under the latent policy to obtain a PAC upper bound on the bisimulation distance between the original and latent models, further bounding their value difference. Additionally, those bounds are also useful when model checking properties: when a property is verified in the latent model, one is now able to lift the values of the property to the original, unknown environment with PAC guarantees.

Now that we can formally link the original environment and the latent model, we are ready to turn to the problem of learning a latent space model and a latent policy from simulations.

2 Variational MDP

In this section, we present a framework based on *variational auto-encoders* (VAE, Kingma and Welling, 2014) that allows us to learn a discrete latent space model of $\mathcal{M}$ from trajectories collected by the agent through the execution of a (learned) RL policy π within the environment. Concretely, *our goal is to learn a discrete latent space model $\langle \bar{\mathcal{M}}, \phi, \psi \rangle$* . To do so, we propose to learn the parameters $\langle \iota, \theta \rangle$ of an *encoder* $\Phi(\cdot; \iota)$ and a *behavioral model* $\Psi(\cdot; \theta)$ from which one can retrieve

- (i) the embedding and lifting functions $\phi(\cdot; \iota)$ and $\psi(\cdot; \iota, \theta)$
- (ii) the latent MDP components $\bar{\mathbf{P}}(\cdot; \theta)$, $\bar{\mathcal{R}}(\cdot; \theta)$, and $\bar{\ell}(\cdot; \theta)$, as well as
- (iii) a latent stationary policy $\bar{\pi}(\cdot; \theta)$.

The goal is to learn (i) a discrete representation of $\mathcal{S}$ and $\mathcal{A}$ (ii) to mimic the behaviors of the original MDP over the induced latent spaces, and (iii) to *distill* π into $\bar{\pi}$.

Along the following sections, we fix an MDP $\mathcal{M}$ and a stationary policy π for $\mathcal{M}$.⁵ Since we are interested in learning probability measures, we learn them through parameterized density functions. Henceforth, for a probability measure $\mathcal{D}$, we write $p_{\mathcal{D}}$ for the density function associated with $\mathcal{D}$.

Intuitively, our approach relies on generating finite *traces* (cf. Definition 6) from $\mathcal{M}_{\pi}$, embedding them to the latent space, and learning to reconstruct them from the latent space. To properly reason about finite traces, we define the *distribution over finite traces of $\mathcal{M}$* as follows.

Definition 38: Distribution over finite traces

Let $\mathcal{M}$ be an MDP, π be a policy, and $H \geq 0$ be a finite time horizon. We write $\mathcal{T}_{\pi}^{\mathcal{M}}(H)$ for the *distribution over traces of $\mathcal{M}_{\pi}$ of length H* so that $\hat{\tau} \sim \mathcal{T}_{\pi}^{\mathcal{M}}(H)$ if and only if $\tau \sim \mathbb{P}_{\pi}^{\mathcal{M}}$ where $\hat{\tau}$ is a prefix of τ and $\hat{\tau} = \langle s_{0:H}, a_{0:H-1}, r_{0:H-1}, l_{0:H} \rangle \in \text{Traces}_{\mathcal{M}}(H)$ with $s_0 = s_I$, $r_t = \mathcal{R}(s_t, a_t)$, $l_t = \ell(s_t)$ for all $t < H$, and $l_H = \ell(s_H)$ (cf. Definition 6).

Precisely, for a given horizon $H \geq 0$, our goal is to optimize the parameters θ of the model $\Psi \in \Delta(\text{Traces}_{\mathcal{M}})$ via the following optimization problem:

$$\min_{\theta} D(\mathcal{T}_{\pi}^{\mathcal{M}}(H), \Psi(\cdot; \theta))$$

where D is a discrepancy measure. In this section, we stick to the (scaled) Kullback-Leibler divergence (cf. Definition 32), i.e., $\min_{\theta} \frac{1}{H} D_{\text{KL}}(\mathcal{T}_{\pi}^{\mathcal{M}}(H) \parallel \Psi(\cdot; \theta))$, where $1/H$ is a scaling factor to ensure that the divergence does not blow up with the horizon H . Minimizing the KL divergence is equivalent to maximizing the log-likelihood of the traces generated by the behavioral model [Murphy, 2023, Section 24.2]:

$$\begin{aligned} & D_{\text{KL}}(\mathcal{T}_{\pi}^{\mathcal{M}}(H) \parallel \Psi(\cdot; \theta)) \\ &= \int_{\text{Traces}_{\mathcal{M}_{\pi}}(H)} \log \left(\frac{\mathcal{T}_{\pi}^{\mathcal{M}}(d\hat{\tau} \mid H)}{\Psi(d\hat{\tau}; \theta)} \right) \mathcal{T}_{\pi}^{\mathcal{M}}(d\hat{\tau} \mid H) \\ &= \int_{\text{Traces}_{\mathcal{M}_{\pi}}(H)} \log \mathcal{T}_{\pi}^{\mathcal{M}}(d\hat{\tau} \mid H) \cdot \mathcal{T}_{\pi}^{\mathcal{M}}(d\hat{\tau} \mid H) - \int_{\text{Traces}_{\mathcal{M}_{\pi}}(H)} \log \Psi(d\hat{\tau}; \theta) \cdot \mathcal{T}_{\pi}^{\mathcal{M}}(d\hat{\tau} \mid H) \\ &= -h(\mathcal{T}_{\pi}^{\mathcal{M}}(H)) - \mathbb{E}_{\hat{\tau} \sim \mathcal{T}_{\pi}^{\mathcal{M}}(H)} [\log p_{\Psi}(\hat{\tau}; \theta)], \end{aligned}$$

where $h(\mathcal{T}_{\pi}^{\mathcal{M}}(H))$ denotes the entropy of $\mathcal{T}_{\pi}^{\mathcal{M}}(H)$. Since the entropy is a constant and does not depend on θ , minimizing the KL divergence $\min_{\theta} D_{\text{KL}}(\mathcal{T}_{\pi}^{\mathcal{M}}(H) \parallel \Psi(\cdot; \theta))$

⁵The approach can be trivially extended to non-stationary policies by considering π as a Mealy machine. Then, states which are embedded to the latent space belong to the product of the state space of the original MDP and the one of the Mealy machine.

is equivalent to maximizing the log-likelihood of the traces generated, i.e., $\max_{\theta} \mathbb{E}_{\hat{\tau} \sim \mathcal{T}_{\pi}^{\mathcal{M}}(H)} [\log p_{\Psi}(\hat{\tau}; \theta)]$. We now assume that Ψ is a *latent variable model* over a latent space $\mathcal{Z}$, where a sequence of latent variables $z_{0:H}$ is generated from a latent MDP $\bar{\mathcal{M}}$, and then *decoded* into a trace $\hat{\tau} = \langle s_{0:H}, a_{0:H-1}, r_{0:H-1}, l_{0:H} \rangle$. In the following, we distinguish the case where we only learn $\mathcal{Z} = \bar{\mathcal{S}}$ when $\bar{\mathcal{A}} = \mathcal{A}$ (in that case, $\mathcal{A}$ is assumed to be discrete), and the one where we additionally need to discretize the set of actions and learn $\bar{\mathcal{A}}$ when $\mathcal{Z} = \bar{\mathcal{S}} \times \bar{\mathcal{A}}$.

Remark 20: General latent space

Note that for now, we do not necessarily restrict our attention to the case where $\mathcal{Z}$ is a finite set. We first present a general theory for continuous latent variables and we will discuss the case of discrete latent variables in Section 2.4, where our approach is particularly well-suited.

By the law of total probability, for any trace $\hat{\tau} \in \text{Traces}_{\mathcal{M}_{\pi}}(H)$, we have

$$p_{\Psi}(\hat{\tau}; \theta) = \int_{\text{Traj}_{\bar{\mathcal{M}}} } p_{\Psi}(\hat{\tau} \mid z_{0:H}; \theta) \cdot \bar{\mathbf{p}}(z_{0:H}; \theta) \, dz_{0:\infty}, \quad (3.8)$$

where $z_t \in \mathcal{Z}$ for all $t \geq 0$ and $\bar{\mathbf{p}}$ is the density of the latent transition function $\bar{\mathbf{P}}_{\pi}$ of $\bar{\mathcal{M}}_{\pi}$, which is defined as

$$\begin{aligned} \bar{\mathbf{p}}(\bar{s}_{0:H}; \theta) &= \prod_{t=0}^{H-1} \mathbb{E}_{a \sim \bar{\pi}(\cdot \mid \bar{s}_t; \theta)} \bar{\mathbf{p}}(\bar{s}_{t+1} \mid \bar{s}_t, a; \theta) && \text{if } \mathcal{Z} = \bar{\mathcal{S}}, \text{ and} \\ \bar{\mathbf{p}}(\bar{s}_{0:H}, \bar{a}_{0:H-1}; \theta) &= \prod_{t=0}^{H-1} p_{\pi}(\bar{a}_t \mid \bar{s}_t; \theta) \cdot \bar{\mathbf{p}}(\bar{s}_{t+1} \mid \bar{s}_t, \bar{a}_t; \theta) && \text{if } \mathcal{Z} = \bar{\mathcal{S}} \times \bar{\mathcal{A}}. \end{aligned}$$

2.1 The Evidence Lower Bound

Due to the integration over the whole trajectory space, optimizing $\mathbb{E}_{\hat{\tau} \sim \mathcal{T}_{\pi}^{\mathcal{M}}(H)} [\log p_{\Psi}(\hat{\tau})]$ with respect to Equation 3.8 is typically intractable [Kingma and Welling, 2014]. Intuitively, estimating Equation 3.8 by drawing random trajectories from $\bar{\mathbf{P}}_{\pi}$ may necessitate an extensive number of samples to approximate $p_{\Psi}(\hat{\tau}) = \mathbb{E}_{z_{0:H} \sim \bar{\mathbf{P}}_{\pi}} p_{\Psi}(\hat{\tau} \mid z_{0:H})$. This is because for randomly drawn trajectories $z_{0:H}$ in $\bar{\mathcal{M}}$, the likelihood of the trace $p_{\Psi}(\hat{\tau} \mid z_{0:H})$ is typically very low. The key concept behind *variational inference* is to focus on samples $z_{0:H}$ that are likely to have generated the trace $\hat{\tau}$, and compute $p_{\Psi}(\hat{\tau} \mid z_{0:H})$ only for those. In essence, we seek the *posterior distribution* $p_{\Psi}(z_{0:H} \mid \hat{\tau})$, but the latter is again intractable to estimate, due to the denominator in Bayes' rule:

$$p_{\Psi}(z_{0:H} \mid \hat{\tau}) = \frac{p_{\Psi}(\hat{\tau} \mid z_{0:H}) \bar{\mathbf{p}}(z_{0:H})}{\int p_{\Psi}(\hat{\tau} \mid z_{0:H}) \bar{\mathbf{p}}(z_{0:H}) \, dz_{0:H}}.$$

To overcome this, we use an *encoder* $\Phi: \text{Traces}_{\mathcal{M}}(H) \rightarrow \Delta(\text{Traj}_{\overline{\mathcal{M}}})$ which aims at emulating the posterior distribution. To do so, we consider minimizing the divergence between $\Phi(\cdot \mid \hat{\tau})$ and $\Psi(\cdot \mid \hat{\tau})$, which allows to set up a *lower bound* on the log-likelihood of produced traces, often referred to as *evidence lower bound* (ELBO, Hoffman et al., 2013):

$$\begin{aligned}
 & D_{\text{KL}}(\Phi(\cdot \mid \hat{\tau}) \parallel \Psi(\cdot \mid \hat{\tau})) \\
 &= \mathbb{E}_{z_{0:H} \sim \Phi(\cdot \mid \hat{\tau})} [\log p_{\Phi}(z_{0:H} \mid \hat{\tau}) - p_{\Psi}(z_{0:H} \mid \hat{\tau})] \\
 &= \mathbb{E}_{z_{0:H} \sim \Phi(\cdot \mid \hat{\tau})} [\log p_{\Phi}(z_{0:H} \mid \hat{\tau}) - \log p_{\Psi}(\hat{\tau} \mid z_{0:H}) - \log \bar{\mathbf{p}}(z_{0:H})] + \log p_{\Psi}(\hat{\tau}) \\
 &\iff \log p_{\Psi}(\hat{\tau}) - D_{\text{KL}}(\Phi(\cdot \mid \hat{\tau}) \parallel \Psi(\cdot \mid \hat{\tau})) \\
 &= \mathbb{E}_{z_{0:H} \sim \Phi(\cdot \mid \hat{\tau})} [\log p_{\Psi}(\hat{\tau} \mid z_{0:H}) + \log \bar{\mathbf{p}}(z_{0:H}) - \log p_{\Phi}(z_{0:H} \mid \hat{\tau})] \\
 &= \mathbb{E}_{z_{0:H} \sim \Phi(\cdot \mid \hat{\tau})} [\log p_{\Psi}(\hat{\tau} \mid z_{0:H})] - D_{\text{KL}}(\Phi(\cdot \mid \hat{\tau}) \parallel \mathbf{P}_{\bar{\pi}}).
 \end{aligned}$$

Then, for a given time horizon H and considering the distribution over traces, the (scaled) ELBO is given by

$$\frac{1}{H} \mathbb{E}_{\hat{\tau} \sim \mathcal{T}_{\pi}^{\mathcal{M}}(H)} [\log p_{\Psi}(\hat{\tau})] \geq \frac{1}{H} \mathbb{E}_{\hat{\tau} \sim \mathcal{T}_{\pi}^{\mathcal{M}}(H)} \left[\mathbb{E}_{z_{0:H} \sim \Phi(\cdot \mid \hat{\tau})} [\log p_{\Psi}(\hat{\tau} \mid z_{0:H})] - D_{\text{KL}}(\Phi(\cdot \mid \hat{\tau}) \parallel \mathbf{P}_{\bar{\pi}}) \right]. \quad (3.9)$$

Notice that the ELBO can be subdivided into two terms: a *reconstruction* $\mathbb{E}_{z_{0:H} \sim \Phi(\cdot \mid \hat{\tau})} [\log p_{\Psi}(\hat{\tau} \mid z_{0:H})]$ and a *regularization term* $D_{\text{KL}}(\Phi(\cdot \mid \hat{\tau}) \parallel \mathbf{P}_{\bar{\pi}})$. In information theory, the negative reconstruction term is often referred to as the *distortion* and the regularization term as the *rate* of the variational model [Alemi et al., 2018].

2.2 The Variational Architecture

Since we deal with a latent model whose variables are generated by the latent MC $\overline{\mathcal{M}}_{\bar{\pi}}$, we restrict our attention to Markovian measures to model both Φ and Ψ . Precisely, the different components of the latent space model are depicted in Figure 3.4 and are retrieved as follows.

Encoders. We learn a *stochastic* embedding function $\phi(\cdot; \iota): \mathcal{S} \rightarrow \Delta(\overline{\mathcal{S}})$ and a *action encoder* $\phi^{\mathcal{A}}(\cdot; \iota): \overline{\mathcal{S}} \times \mathcal{A} \rightarrow \Delta(\overline{\mathcal{A}})$ so that:

- (i) $p_{\Phi}(\bar{s}_{0:H} \mid \hat{\tau}) = \prod_{t=0}^H p_{\phi}(\bar{s}_t \mid s_t)$, when $\mathcal{Z} = \overline{\mathcal{S}}$, and
- (ii) $p_{\Phi}(\bar{s}_{0:H}, \bar{a}_{0:H-1} \mid \hat{\tau}) = p_{\phi}(\bar{s}_H \mid s_H) \cdot \prod_{t=0}^{H-1} p_{\phi^{\mathcal{A}}}(\bar{a}_t \mid \bar{s}_t, a_t) \cdot p_{\phi}(\bar{s}_t \mid s_t)$, when $\mathcal{Z} = \overline{\mathcal{S}} \times \overline{\mathcal{A}}$.

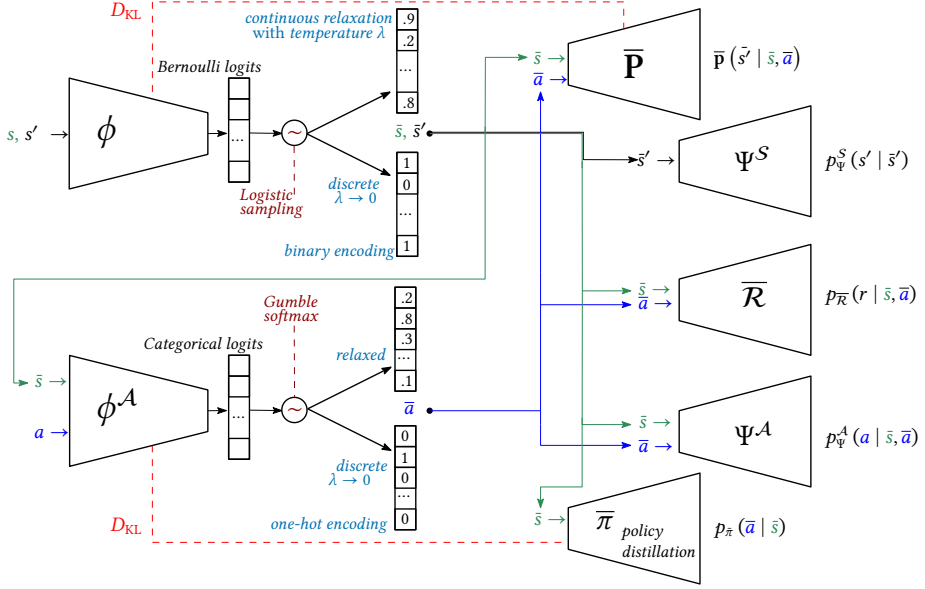


Figure 3.4: Architecture of our variational model. Each component is a neural network.

Decoders. We further learn *stochastic* label and reward functions, as well as state and action *decoders*, all of them allowing decomposing the behavioral model Ψ as

$$\begin{aligned}
 & p_{\Psi}(s_{0:H}, a_{0:H-1}, r_{0:H-1}, l_{0:H} \mid z_{0:H}) \\
 &= p_{\Psi}^S(s_{0:H} \mid \bar{s}_{0:H}) \cdot p_{\Psi}^A(a_{0:H-1} \mid z_{0:H}) \cdot p_{\bar{R}}(r_{0:H-1} \mid \bar{s}_{0:H}, \bar{a}_{0:H-1}) \cdot p_{\bar{l}}(l_{0:H} \mid \bar{s}_{0:H}),
 \end{aligned}$$

where

- (iii) $p_{\Psi}^S(s_{0:H} \mid \bar{s}_{0:H}) = \prod_{t=0}^H p_{\Psi}^S(s_t \mid \bar{s}_t)$;
- (iv) $p_{\Psi}^A(a_{0:H-1} \mid z_{0:H}) = \begin{cases} \prod_{t=0}^{H-1} \bar{\pi}(a_t \mid \bar{s}_t) & \text{if } \mathcal{A} = \bar{\mathcal{A}}, \text{ and} \\ \prod_{t=0}^{H-1} p_{\Psi}^A(a_t \mid \bar{s}_t, \bar{a}_t) & \text{otherwise;} \end{cases}$
- (v) $p_{\bar{R}}(r_{0:H-1} \mid \bar{s}_{0:H}, \bar{a}_{0:H-1}) = \prod_{t=0}^H p_{\bar{R}}(r_t \mid \bar{s}_t, \bar{a}_t)$; and
- (vi) $p_{\bar{l}}(l_{0:H} \mid \bar{s}_{0:H}) = \prod_{t=0}^H p_{\bar{l}}(l_t \mid \bar{s}_t)$.

We call the resulting model a *variational auto-encoded Markov decision process* (VAE-MDP), or simply a variational MDP. Note that model p_{Ψ}^A allows learning a *stochastic*

action lifting function $\psi(\cdot; \iota, \theta) : S \times \bar{\mathcal{A}} \rightarrow \Delta(\mathcal{A})$, via $p_\psi(a \mid s, \bar{a}) = \mathbb{E}_{\bar{s} \sim \phi(\cdot \mid s; \iota)} p_\psi^A(a \mid \bar{s}, \bar{a}; \theta)$. Additionally, in line with Assumption 4, we argue that our state embedding function allows for a *perfect reconstruction of labels*, meaning $\mathbb{E}_{\hat{\tau} \sim \mathcal{T}_\pi^{\mathcal{M}}(H)} [p_{\bar{\ell}}(l_{0:H} \mid \bar{s}_{0:H})] = 1$. From now on, we thus omit the label term. More details on designing a pair of embedding and latent labeling functions that allow this perfect reconstruction are given in Section 2.4.

The deterministic counterpart of stochastic functions ϕ , ψ , and $\bar{\mathcal{R}}$ can be merely obtained by taking the mode of their respective distribution.

2.3 Back to the Local Setting

In this section, we assume that the environment is an episodic process. Then, drawing multiple finite traces $\hat{\tau} \sim \mathcal{T}_\pi^{\mathcal{M}}(H)$ for different horizons $H \geq 0$ can be seen as drawing episodes along infinite traces via a continuous interaction with $\mathcal{M}$ (cf. Chapter 2, Section 2.1). This observation allows us to formulate the ELBO in the local setting and connect it to local losses. To that aim, we first introduce the following technical Lemma.

Lemma 15: Ergodic objective function

Assume that $\mathcal{M}$ is episodic and $S, \mathcal{A}$ are compact spaces. Let ξ_π be the stationary distribution of $\mathcal{M}_\pi$ and $f : S \times \mathcal{A} \times [-R_{\max}, R_{\max}] \times S \rightarrow \mathbb{R}$ be a continuous function. Then,

$$\lim_{H \rightarrow \infty} \frac{1}{H} \mathbb{E}_{\hat{\tau} \sim \mathcal{T}_\pi^{\mathcal{M}}(H)} \left[\sum_{t=0}^{H-1} f(s_t, a_t, r_t, s_{t+1}) \right] = \mathbb{E}_{s, a, r, s' \sim \xi_\pi} f(s, a, r, s'),$$

where each $\hat{\tau}$ has the form $\langle s_{0:H}, a_{0:H-1}, r_{0:H-1}, l_{0:H} \rangle$.

Proof. First, observe that

$$\begin{aligned} & \mathbb{E}_{\hat{\tau} \sim \mathcal{T}_\pi^{\mathcal{M}}(H)} \left[\sum_{t=0}^{H-1} f(s_t, a_t, r_t, s_{t+1}) \right] \\ &= \mathbb{E}_{a_0 \sim \pi(\cdot \mid s_0)} \mathbb{E}_{s_1 \sim \mathbf{P}(\cdot \mid s_0, a_0)} \mathbb{E}_{a_1 \sim \pi(\cdot \mid s_1)} \cdots \mathbb{E}_{s_H \sim \mathbf{P}(\cdot \mid s_{H-1}, a_{H-1})} \left[\sum_{t=0}^{H-1} f(s_t, a_t, \mathcal{R}(s_t, a_t), s_{t+1}) \right] \\ &= \sum_{t=0}^{H-1} \mathbb{E}_{a_0 \sim \pi(\cdot \mid s_0)} \mathbb{E}_{s_1 \sim \mathbf{P}(\cdot \mid s_0, a_0)} \mathbb{E}_{a_1 \sim \pi(\cdot \mid s_1)} \cdots \mathbb{E}_{s_H \sim \mathbf{P}(\cdot \mid s_{H-1}, a_{H-1})} f(s_t, a_t, \mathcal{R}(s_t, a_t), s_{t+1}) \\ &= \sum_{t=0}^{H-1} \mathbb{E}_{\hat{\tau} \sim \mathcal{T}_\pi^{\mathcal{M}}(t+1)} f(s_t, a_t, r_t, s_{t+1}). \end{aligned}$$

Second, let $t \geq 0$, $s, s' \in \mathcal{S}$, $a \in \mathcal{A}$, and $r = \mathcal{R}(s, a)$, also observe that

$$\begin{aligned} & \int_{\mathcal{S}} f(s, a, r, s') \, d\xi_{\pi}^t(s \mid s_t) \\ &= \int_{\mathcal{S}} \int_{\text{Traces}_{\mathcal{M}_{\pi}}(t)} f(s, a, r, s') \cdot \delta_{s_t}(s) \, d\mathbb{P}_{\pi}(\text{Cyl}(s_{0:t}, a_{0:t-1})) \, ds \\ &= \int_{\text{Traces}_{\mathcal{M}_{\pi}}(t)} f(s_t, a, r, s') \, d\mathbb{P}_{\pi}(\text{Cyl}(s_{0:t}, a_{0:t-1})), \end{aligned}$$

where $\delta_{s_t}(s)$ is the Dirac measure assigning all its mass at s_t and $\text{Cyl}(\tau)$ is the cylinder spanned by the finite prefix τ (cf. Theorem 1). Therefore, we have

$$\begin{aligned} & \lim_{H \rightarrow \infty} \frac{1}{H} \mathbb{E}_{\hat{\tau} \sim \mathcal{T}_{\pi}^{\mathcal{M}}(H)} \left[\sum_{t=0}^{H-1} f(s_t, a_t, r_t, s_{t+1}) \right] \\ &= \lim_{H \rightarrow \infty} \frac{1}{H} \sum_{t=0}^{H-1} \mathbb{E}_{\hat{\tau} \sim \mathcal{T}_{\pi}^{\mathcal{M}}(t+1)} f(s_t, a_t, r_t, s_{t+1}) \\ &= \lim_{H \rightarrow \infty} \frac{1}{H} \sum_{t=0}^{H-1} \int_{\text{Traces}_{\mathcal{M}_{\pi}}(t+1)} f(s_t, a_t, r_t, s_{t+1}) \, d\mathbb{P}_{\pi}(\text{Cyl}(s_{0:t+1}, a_{0:t})) \\ &= \lim_{H \rightarrow \infty} \frac{1}{H} \sum_{t=0}^{H-1} \int_{\mathcal{A}} \int_{\mathcal{S}} \int_{\text{Traces}_{\mathcal{M}_{\pi}}(t)} f(s_t, a, \mathcal{R}(s_t, a), s') \\ & \quad d\mathbb{P}_{\pi}(\text{Cyl}(s_{0:t+1}, a_{0:t})) \, d\pi(a \mid s) \, d\mathbb{P}(s' \mid s, a) \\ &= \lim_{H \rightarrow \infty} \frac{1}{H} \sum_{t=0}^{H-1} \int_{\mathcal{S}} \int_{\mathcal{A}} \int_{\mathcal{S}} f(s, a, \mathcal{R}(s, a), s') \, d\xi_{\pi}^t(s \mid s_t) \, d\pi(a \mid s) \, d\mathbb{P}(s' \mid s, a) \\ &= \lim_{H \rightarrow \infty} \int_{\mathcal{S}} \frac{1}{H} \sum_{t=0}^{H-1} d\xi_{\pi}^t(s \mid s_t) \int_{\mathcal{A}} \int_{\mathcal{S}} f(s, a, \mathcal{R}(s, a), s') \, d\pi(a \mid s) \, d\mathbb{P}(s' \mid s, a). \end{aligned}$$

By definition of the stationary distribution ξ_{π} (Chapter 2, Section 1.4, Equation 2.3) and since $\mathcal{M}$ is ergodic, then $\xi_{\pi}(E) = \lim_{H \rightarrow \infty} \frac{1}{H} \sum_{t=0}^{H-1} \xi_{\pi}^t(E \mid s)$ almost surely (with respect to $\mathbb{P}_{\pi}^{\mathcal{M}}$) for all $s \in \mathcal{S}$, $E \in \Sigma(\mathcal{S})$ [Baier and Katoen, 2008; Asmussen and Glynn, 2011].

The *Portmanteau theorem* [Dudley, 2002] tells us that given a complete, separable metric space $\mathcal{X}$, $(\nu_n)_{n \geq 0}$, and $\nu \in \Delta(\mathcal{X})$, the following statements are equivalent:

- (i) ν_n weakly converges to ν as $n \rightarrow \infty$;
- (ii) for any continuous subset $E \in \Sigma(\mathcal{X})$, $\lim_{n \rightarrow \infty} \nu_n(E) = \nu(E)$;
- (iii) for any bounded and continuous function f , $\lim_{n \rightarrow \infty} \int_{\mathcal{X}} f(\cdot) \, d\nu_n = \int_{\mathcal{X}} f(\cdot) \, d\nu$.

By definition, the set of images of $\mathcal{R}$ is a compact space, thus f has compact support. Almost sure convergence implies weak convergence [Geng, 2023], thus by the Portmanteau theorem,

$$\begin{aligned} & \lim_{H \rightarrow \infty} \int_S \frac{1}{H} \sum_{t=0}^{H-1} d\xi_\pi^t(s \mid s_I) \int_{\mathcal{A}} \int_S f(s, a, \mathcal{R}(s, a), s') d\pi(a \mid s) d\mathbf{P}(s' \mid s, a) \\ &= \int_S d\xi_\pi(s) \int_{\mathcal{A}} \int_S f(s, a, \mathcal{R}(s, a), s') d\pi(a \mid s) d\mathbf{P}(s' \mid s, a) \\ &= \mathbb{E}_{s, a, r, s' \sim \xi_\pi} f(s, a, r, s'). \end{aligned}$$

□

Theorem 26: Local ELBO

Assume that $\mathcal{M}$ is an episodic process, S and $\mathcal{A}$ are compact spaces, and that all the parameterized densities of our variational model are continuous. Then,

$$\lim_{H \rightarrow \infty} \frac{1}{H} \mathbb{E}_{\substack{\hat{\tau} \sim \mathcal{T}_\pi^{\mathcal{A}}(H) \\ z_0:H \sim \Phi(\cdot \mid \hat{\tau})}} [\log p_\Psi(\hat{\tau} \mid z_0:H) - D_{\text{KL}}(\Phi(\cdot \mid \hat{\tau}) \parallel \Psi(\cdot \mid \hat{\tau}))] = -(\mathbf{D} + \mathbf{R})$$

where $\mathbf{D}$ and $\mathbf{R}$ denote the *distortion* and the *rate* of the variational model, respectively defined as

$$\mathbf{D} = - \begin{cases} \mathbb{E}_{\substack{s, a, r, s' \sim \xi_\pi \\ \bar{s}, \bar{s}' \sim \phi(\cdot \mid s, s')}} [\log p_\Psi^S(s' \mid \bar{s}') + \log p_{\bar{\pi}}(a \mid \bar{s}) + \log p_{\bar{r}}(r \mid \bar{s}, a)] & \text{if } \bar{\mathcal{A}} = \mathcal{A}, \\ \mathbb{E}_{\substack{s, a, r, s' \sim \xi_\pi \\ \bar{s}, \bar{s}' \sim \phi(\cdot \mid s, s') \\ \bar{a} \sim \phi^{\mathcal{A}}(\cdot \mid \bar{s}, a)}} [\log p_\Psi^S(s' \mid \bar{s}') + \log p_\Psi^{\mathcal{A}}(a \mid \bar{s}, \bar{a}) + \log p_{\bar{r}}(r \mid \bar{s}, \bar{a})] & \text{else; and} \end{cases}$$

$$\mathbf{R} = \begin{cases} \mathbb{E}_{\substack{s, a, s' \sim \xi_\pi \\ \bar{s} \sim \phi(\cdot \mid s)}} D_{\text{KL}}(\phi(\cdot \mid s') \parallel \bar{\mathbf{P}}_{\bar{\pi}}(\cdot \mid \bar{s})) & \text{if } \bar{\mathcal{A}} = \mathcal{A}, \\ \mathbb{E}_{\substack{s, a, s' \sim \xi_\pi \\ \bar{s} \sim \phi(\cdot \mid s) \\ \bar{a} \sim \phi^{\mathcal{A}}(\cdot \mid \bar{s}, a)}} [D_{\text{KL}}(\phi(\cdot \mid s') \parallel \bar{\mathbf{P}}(\cdot \mid \bar{s}, \bar{a})) + D_{\text{KL}}(\phi^{\mathcal{A}}(\cdot \mid \bar{s}, a) \parallel \bar{\pi}(\cdot \mid \bar{s}))] & \text{else,} \end{cases}$$

where $p_\phi(\bar{s}, \bar{s}' \mid s, s') = p_\phi(\bar{s} \mid s) \cdot p_\phi(\bar{s}' \mid s')$.

Proof. Substituting

$$f(s, a, r, s') \quad \text{to} \quad \mathbb{E}_{\bar{s}, \bar{s}' \sim \phi(\cdot \mid s, s')} g(\langle s, a, r, s' \rangle, \langle \bar{s}, \bar{s}' \rangle)$$

in Lemma 15 with

$$\begin{aligned}
& g(\langle s, a, r, s' \rangle, \langle \bar{s}, \bar{s}' \rangle) \\
&= \begin{cases} \log(p_{\Psi}^S(s' | \bar{s}') \cdot p_{\bar{R}}(r | \bar{s}, a) \cdot p_{\bar{\pi}}(a | \bar{s})) - D_{\text{KL}}(\phi(\cdot | s') \| \bar{\mathbf{P}}_{\bar{\pi}}(\cdot | \bar{s})) & \text{if } \bar{\mathcal{A}} = \mathcal{A}, \text{ and} \\ \mathbb{E}_{\bar{a} \sim \phi^A(\cdot | \bar{s}, a)} \left[\log(p_{\Psi}^S(s' | \bar{s}') \cdot p_{\Psi}^A(a | \bar{s}, \bar{a}) \cdot p_{\bar{\Psi}}^{\bar{R}}(r | \bar{s}, \bar{a})) \right. & \text{otherwise} \\ \quad \left. - D_{\text{KL}}(\phi(\cdot | s') \| \bar{\mathbf{P}}(\cdot | \bar{s}, \bar{a})) \right] - D_{\text{KL}}(\phi^A(\cdot | \bar{s}, a) \| \bar{\pi}(\cdot | \bar{s})) & \end{cases}
\end{aligned}$$

yields the result. $\square$

The compactness assumption of Theorem 26 is very mild and is met in all scenarios relying on a simulator. We further show in Section 2.4 how to ensure that the parameterized densities of our variational model are continuous at training time.

Local loss function. Theorem 26 allows us to re-write the objective function of our variational model as a local loss function:

$$\max_{\iota, \theta} \text{ELBO}(\bar{\mathcal{M}}, \phi, \psi, \bar{\pi}) = - \min_{\iota, \theta} \{ \mathbf{D}_{\iota, \theta} + \mathbf{R}_{\iota, \theta} \}.$$

The optimization of the ELBO allows for an indirect optimization of the local losses through their variational versions: (i) $L_{\mathcal{R}}$ via the log-likelihood of rewards produced, and (ii) $L_{\mathbf{P}}^{\uparrow}$ where we change the Wasserstein term to the KL divergence. Note that this last change means that we do not necessarily obtain the theoretical abstraction quality guarantees via the optimization of the ELBO. Nevertheless, our experiments (Section 2.6) indicate that the KL divergence is a good proxy of the Wasserstein distance in practice. In particular, in the discrete setting, Wasserstein matches TV and one can relate the proxy with the original metric using the Pinsker’s inequality (cf. Theorem 18).

2.4 Variational Distributions

Ultimately, we aim at learning discrete latent spaces $\bar{\mathcal{S}}$ and $\bar{\mathcal{A}}$ to fulfill Assumption 3 and leverage the guarantees derived in the previous section, to further enable the verification of the latent model via model checking algorithms. We thus seek discrete distributions for ϕ , ϕ^A , $\bar{\mathbf{P}}$, and $\bar{\pi}$. Three main challenges arise when the latent space is discrete:

- (i) gradient descent is not applicable to learn ι and θ in the discrete setting since this imply the discontinuity of $\bar{\mathcal{S}}$ and $\bar{\mathcal{A}}$,
- (ii) the density functions must further be differentiable (and thus continuous) to apply gradient descent, and

- (iii) sampling from the encoder to compute the ELBO must also be a differentiable operation.

We overcome these by using *continuous relaxation of Bernoulli distributions* to learn a binary representation of the latent states, and the *Gumbel softmax trick* for the latent action space [Jang et al., 2017; Maddison et al., 2017]. Intuitively, the trick relies on a temperature parameter, $\lambda \in (0, 1)$. The latter controls the continuity of the latent space learned. The technique yields *re-parameterizable* and convex densities, which are compliant with stochastic gradient descent. The zero-temperature limit (i.e., passing from the continuous to the discrete setting) is theoretically enabled via simple rounding of continuous random variables. Furthermore, the logits of discrete densities are guaranteed to be identical to those of their relaxed counterparts.

Latent Distributions: Continuous Relaxations of Discrete Distributions

The reparameterization trick. Sampling from a distribution P may sometimes involve two steps: (i) sampling from a simple, *parameter-free distribution* $y \sim Q$, and (ii) applying a *deterministic transformation* f to the sample, resulting in a new sample $x = f(y)$ with distribution P . A notable example is the *normal distribution*. Drawing a sample x from the normal distribution $\mathcal{N}(\cdot; \mu, \sigma^2)$ with mean μ and variance σ^2 can be realized by drawing a random noise y from the standard normal distribution $\mathcal{N}(\cdot; 0, 1)$ and applying the transformation $x = \mu + \sigma \cdot y$.

This approach, known as the *reparameterization trick*, shifts the dependence of P on its parameters to the parameters of the deterministic function f . Consequently, estimating the gradients of sampling from a reparameterizable distribution simplifies to estimating the gradients with respect to the parameters of a deterministic function.

Latent action space. We learn discrete latent distributions for ϕ^A and $\bar{\pi}$ by inferring the *logits of continuous relaxation of discrete distributions*, through the *Gumbel softmax trick*. Concretely, let $n = |\bar{A}|$ and $\bar{A}$ be enumerated as $\{\bar{a}_1, \dots, \bar{a}_n\}$. Given some vector $\alpha \in \mathbb{R}^n$, the categorical distribution $\text{Categorical}(\alpha)$ has for probability measure $P(\bar{a}_i) = \exp(\alpha_i) / \sum_{k=1}^n \exp(\alpha_k) \in \Delta(\bar{A})$. We call α the *logits* of the distribution.

Sampling from a categorical distribution can be realized by reparameterizing the distribution with the *Gumbel* distribution, whose density is merely given by $p_G(x) = \exp(-x - \exp(-x))$. In turn, sampling from the Gumbel distribution is also fully reparameterizable: one simply needs to sample a random noise uniformly in $[0, 1]$ and then apply the function $-\log \circ -\log$ on this noise to obtain a Gumbel, i.e., $g \sim \text{Gumbel} \equiv \epsilon \sim \mathcal{U}(0, 1)$ and $g = -\log(-\log \epsilon)$. Drawing $\bar{a}_i \sim \text{Categorical}(\alpha)$ is equivalent to applying an arg max operation to the sum of *Gumbel samples* and the logits, i.e., $i = \arg \max_k \alpha_k + g_k$, where $g_k \sim \text{Gumbel}$ for all $1 \leq k \leq n$ (cf. Figure 3.5a for

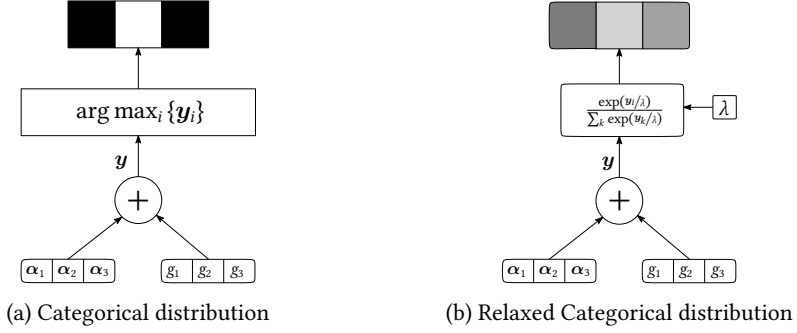


Figure 3.5: The Gumbel softmax trick (figure from Maddison et al., 2017).

an illustration). The sampling operation being fully reparameterizable here, $\arg \max$ is *however not differentiable*, which prevents its optimization through gradient descent. The solution proposed by Jang et al.; Maddison et al. is to replace the $\arg \max$ operation by a *softmax* function with temperature parameter $\lambda \in (0, 1/(n-1)]$ (Figure 3.5b). Concretely, $\mathbf{x} \in [0, 1]^n$ has a *relaxed discrete distribution* $\text{RelaxedCategorical}(\boldsymbol{\alpha}, \lambda)$ with logits $\boldsymbol{\alpha}$ and temperature λ means that

- (a) drawing $\mathbf{x}$ from $\text{RelaxedCategorical}(\boldsymbol{\alpha}, \lambda)$ if equivalent to drawing a uniform noise $\epsilon_k \sim \mathcal{U}(0, 1)$ and computing a Gumbel $g_k = -\log(-\log \epsilon_k)$ so that

$$\mathbf{x}_k = \frac{\exp(\alpha_k + g_k/\lambda)}{\sum_{i=1}^n \exp(\alpha_i + g_i/\lambda)}$$

for all $1 \leq k \leq n$;

- (b) $\lim_{\lambda \rightarrow 0} \text{RelaxedCategorical}(\boldsymbol{\alpha}, \lambda) = \text{Categorical}(\boldsymbol{\alpha})$, which further means that the probability that $\mathbf{x}_k = 1$ is $\exp \alpha_k / \sum_{i=1}^n \exp \alpha_i$ as $\lambda \rightarrow 0$; and

- (c) $p_{\boldsymbol{\alpha}, \lambda}(\mathbf{x})$ is log-convex in $\mathbf{x}$, where $p_{\boldsymbol{\alpha}, \lambda}$ is the density of $\text{RelaxedCategorical}(\boldsymbol{\alpha}, \lambda)$.

In practice, we train the neural networks representing $\phi^{\mathcal{A}}$ and $\bar{\pi}$ to infer the logits $\boldsymbol{\alpha}$. We may also anneal λ from $(n-1)^{-1}$ to 0 during training while using discrete distributions at evaluation time. As prescribed by Maddison et al. [2017], we use two different temperature parameters, one for $\phi^{\mathcal{A}}$ and another one for $\bar{\pi}$.

Latent state space. To allow learning a large, discrete latent state space with compact representation, we assume that the latent states have a binary representation. We further see each bit of their encoding as a *Bernoulli* sample. Bernoulli distributions are a special

case of categorical ones, where $n = 1$. The measure of a Bernoulli distribution with logit $\alpha \in \mathbb{R}$ is $P(x) = \sigma(\alpha)$ whenever $x = 1$ and $1 - \sigma(\alpha)$ when $x = 0$, where $x \in \{0, 1\}$ and σ is the *sigmoid function*, given by $\sigma(x) = \exp(x) / (\exp(x) + 1) = 1 / (1 + \exp(-x))$.

For the particular case of Bernoulli distributions, the Gumbel reparameterization trick merely reduces to sampling from a *logistic distribution*, whose density is given by $p_{\mu,s}(x) = \exp(\mu - x/s) / (s(1 + \exp(\mu - x/s))^2)$ where $\mu \in \mathbb{R}$ is the *location* and $s > 0$ the *scale* of the distribution. Interestingly, as the normal distribution, the logistic distribution is part of the *location-scale family* and is thus fully reparameterizable: sampling from a logistic distribution $l \sim \text{Logistic}(\mu, s)$ can be realized by drawing a noise $\epsilon_l \sim \text{Logistic}(0, 1)$ and applying the transformation $l = \mu + s \cdot \epsilon_l$. In turn, sampling from a Bernoulli distribution with logit α can be realized by drawing a sample $l \sim \text{Logistic}(\alpha, 1)$ and applying the transformation $x = \text{round}(\sigma(l))$, where *round* denotes the *rounding* operation. Again, rounding is not differentiable, so we consider instead a *relaxed* version of this operation: one simply needs to trade the rounding operation to a scaling temperature $\lambda \in (0, 1)$ so that annealing the temperature to zero allows retrieving the rounding operation. Concretely, x has a *relaxed Bernoulli distribution* $\text{RelaxedBernoulli}(\alpha, \lambda)$ with logit α and temperature parameter λ means that

- (a) drawing x from $\text{RelaxedBernoulli}(\alpha, \lambda)$ is equivalent to drawing a logistic sample l at location α/λ and scaled to $1/\lambda$, i.e., $\epsilon_l \sim \text{Logistic}(0, 1)$ and $l = \alpha + \epsilon_l/\lambda$, then applying a sigmoid to the resulting sample $x = \sigma(l)$;
- (b) $\lim_{\lambda \rightarrow 0} \text{RelaxedBernoulli}(\alpha, \lambda) = \text{Bernoulli}(\alpha)$, which further means that the probability that $x = 1$ as the temperature goes to zero is given by $\sigma(\alpha)$; and
- (c) $p_{\alpha,\lambda}(x)$ is log-convex in x , where $p_{\alpha,\lambda}$ is the density of $\text{RelaxedBernoulli}(\alpha, \lambda)$.

In practice, we train ϕ and $\bar{\mathbf{P}}$ to infer and generate $\log_2 |\bar{\mathbf{S}}|$ logits, and we anneal λ to 0 during training while using Bernoulli distributions at evaluation time. Again, as suggested by Maddison et al. [2017], we use two different temperature parameters, one for ϕ and one for $\bar{\mathbf{P}}$.

Remark 21: Tuning the temperature: the variance vs. bias trade-off

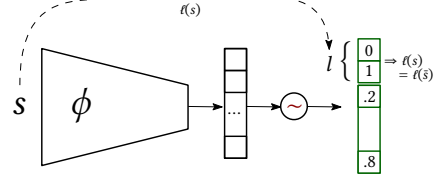
The temperature parameter λ controls the continuity of the distributions. Discrete distributions are thus retrieved in the zero temperature limit. Intuitively, higher temperature produces *low-variance but biased^a gradients*, while lowering the temperature to zero *increases the variance but ends up producing unbiased gradients*.

^awith respect to the target discrete distribution.

Lossless Encoding of the Labels

Given any trace $\langle s_{0:H}, a_{0:H-1}, r_{0:H-1}, l_{0:H} \rangle$ drawn in $\mathcal{M}_\pi$, we enable $\log p_{\bar{\ell}}(l_{0:H} | \bar{s}_{0:H}) = 0$ by designing the embedding function ϕ to *linearly* encode $\ell(s_t) = l_t$ into each $\bar{s}_t$, for all $0 \leq t \leq H$. To do so, since **AP** is assumed finite, we allocate $n = |\mathbf{AP}|$ bits in $\bar{S}$ to encode the labels of each state $s \in S$ in binary.

Precisely, assuming **AP** is enumerated as $\{a_1, \dots, a_n\}$, $\ell(s)$ can be encoded into a vector of n bits $\mathbf{l} \in \{0, 1\}^n$ so that $\mathbf{l}_i = 1$ iff $a_i \in \ell(s)$. Then, $\phi(\bar{s} | s) > 0$ implies $\ell(s) = \bar{\ell}(\bar{s})$, for all $\bar{s} \in \bar{S}$, satisfying Assumption 4 when ϕ_t is deterministic.



Decoders

We learn Ψ^A , ψ , and $\bar{\mathcal{R}}$ as parameters of multivariate normal distributions, and we infer them through neural networks. This further allows to link each $\bar{s} \in \bar{S}$ to the parameters of $p_{\bar{\Psi}}^S(\cdot | \bar{s})$, which may also be useful for explainability purposes.

2.5 Posterior Collapse

A common issue encountered while optimizing variational models via the ELBO is *posterior collapse*. Intuitively, this results in a degenerate local optimum where the model learns to ignore the latent space. With a discrete encoder, this translates into a deterministic mapping to a single latent state, regardless of the input. From an information-theoretic point of view, optimizing the ELBO gives way to a trade-off between the minimization of **R** and **D**, where the feasible region is a convex set [Alemi et al., 2018]. Posterior collapse occurs when $\mathbf{R} \approx 0$ (*auto-decoding limit*). On the other hand, one can achieve $\mathbf{D} \approx 0$ (*auto-encoding limit*) at the price of a higher rate.

Regularization terms. Various solutions have been proposed in the literature to prevent posterior collapse. They include *entropy regularization* (via $\alpha \in [0, \infty)$, e.g., Burke et al. 2019; Dong et al. 2020) and *KL-scaling* (via $\beta \in [0, 1]$, e.g., Alemi et al. 2018), consisting in changing ELBO to $\langle \alpha, \beta \rangle \cdot \text{ELBO} = -(\mathbf{D} + \beta \cdot \mathbf{R}) + \alpha \cdot h(\Phi)$, where $h(\Phi)$ denotes the entropy of the encoding distribution. Specifically, we measure the entropy of the *marginal* encoder, given by $\Phi(\cdot) = \mathbb{E}_{s \sim \xi_\pi} \phi(\cdot | s)$. Intuitively, this encourages the encoder to learn to make plenty of use of the latent space. The parameter β allows interpolating between auto-encoding and auto-decoding behavior, which is not possible with the standard ELBO objective.

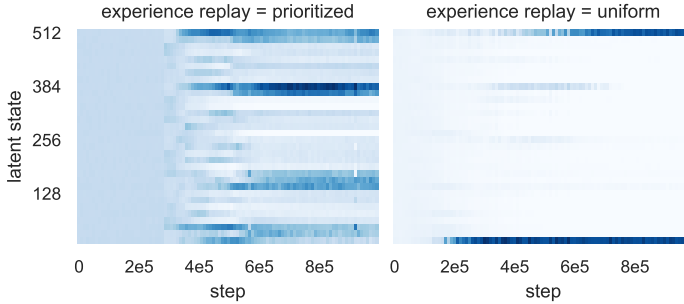


Figure 3.6: Latent space distribution along training steps for the CartPole environment. The intensity of the blue hue corresponds to the frequency of latent states produced by ϕ during training. We compare a bucket-based prioritized against a simple uniform experience replay. The latent space learned when the uniform buffer is used clearly collapses to two distinct restricted regions of the latent space, thereby failing to induce a useful representation of the input states.

A drawback of these methods is that we no longer optimize a lower bound on the log-likelihood of the input while optimizing $\langle \alpha, \beta \rangle \cdot \text{ELBO}$. In practice, setting up annealing schemes for α and β allows to eventually recover ELBO and avoid posterior collapse ($\langle \alpha, \beta \rangle \cdot \text{ELBO}$ matches the original ELBO when $\alpha = 0$ and $\beta = 1$).

Prioritized replay buffers. To enable meaningful use of latent space to represent input state-actions pairs, Φ should learn to (i) fairly spread the input over the entire latent space, and (ii) wisely encode state-action pairs encountered along transitions yielding a poor local ELBO, such transitions being prone to bad representation embedding. To deal with those challenges, we use a *prioritized experience replay buffer* [Schaul et al., 2016] to store transitions and sample them when optimizing ELBO.

Value-based methods (in particular, DQN) often use an experience replay buffer $\mathcal{D}$ to store transitions of the form $\rho = \langle s, a, r, s' \rangle$ encountered along the interaction. Those transitions are then sampled in mini-batches from $\mathcal{D}$ to approximate a loss function of the form $L = \mathbb{E}_{\rho \sim \mathcal{D}} f(\rho)$, where f is a “transition-wise” loss function (e.g., the temporal difference or the mean squared Bellman error). Drawing transitions from the replay buffer allows to circumvent the strong sequential correlation between transitions which necessarily occurs when the agent interacts with the environment (see Chapter 2, Section 2.2 for details). However, vanilla experience replay buffers do not allow to deal with rare transitions that could significantly impact the loss function L . Second, some transitions may be more informative than others, and it may be desirable to sample them

more frequently. *Prioritized* experience replay buffers aim at addressing these issues and are defined as follows.

Definition 39: Prioritized Experience Replay [Schaul et al., 2016]

Prioritized experience replay buffers are finite data structures $\mathcal{D}$ which (i) store each experience ρ with some assigned priority p_ρ , and (ii) produce $\rho \sim \mathcal{D}$ with probability

$$P_{\mathcal{D}}(\rho) = \frac{p_\rho^{\eta_1}}{\sum_{\rho' \in \mathcal{D}} p_{\rho'}^{\eta_1}},$$

where $\eta_1 \in [0, 1]$ is called the *priority exponent*. This allows minimizing $L = \mathbb{E}_{\rho \sim \mathcal{D}} [w_\rho \cdot f(\rho)]$ given an *importance sampling weight* $w_\rho > 0$. The replay buffer $\mathcal{D}$ is *uniform* whenever $\alpha = 0$. Otherwise, when $\alpha > 0$, a bias is introduced because priorities change the real distribution of experiences and consequently the estimates of L . This bias is alleviated by defining the importance sampling weights as $w_\rho = (|\mathcal{D}| \cdot P_{\mathcal{D}}(\rho))^{-\eta_2}$ for each $\rho \in \mathcal{D}$, where $\eta_2 \in [0, 1]$ is another *importance sampling exponent*. In particular, $\eta_2 = 1$ fully corrects the bias.

In practice, we make use of a *prioritized* replay buffer $\mathcal{D}$ to emulate the stationary distribution ξ_π and sample transitions to optimize the ELBO. We introduce two priority functions, namely *bucket-based* and a *loss-based* priorities. Replay buffers using such priorities allow to alleviate the posterior collapse problem, as shown in Figure 3.6. Precisely, draw $\rho = \langle s, a, r, s' \rangle$ from the interaction of the agent executing π within $\mathcal{M}$ and let p_ρ be the priority of this transition. When storing ρ in the replay buffer $\mathcal{D}$, we use one of the following priority functions.

- *Bucket-based priority.* We partition the buffer into $|\bar{\mathcal{S}}|$ *buckets*. Let $t \in \mathbb{N}$ and $b : \bar{\mathcal{S}} \rightarrow \mathbb{N}$ be respectively step and latent state counters. At each step t , draw $\bar{s} \sim \phi(\cdot | s)$ and assign p_ρ to $t/b(\bar{s})$, then increment t and $b(\bar{s})$ of one. This allows ϕ to process states s being infrequently visited under π and learn to fairly distribute $\bar{\mathcal{S}}$.
- *Loss-based priority.* We set p_ρ to its individual loss, which enables learning to improve the representation of states and actions that yield a poor ELBO. Specifically, our goal is to assign p_ρ to its *individual transition loss* f_{ELBO} . Draw $\bar{s}, \bar{s}' \sim \phi(\cdot | s, s')$ and additionally draw $\bar{a} \sim \phi^{\mathcal{A}}(\cdot | \bar{s}, a)$ if $\bar{\mathcal{A}} \neq \mathcal{A}$; otherwise simply set $\bar{a}$ to a . Then f_{ELBO} is given by

$$\begin{aligned} f_{\text{ELBO}}(\langle s, a, r, s' \rangle, \langle \bar{s}, \bar{a}, \bar{s}' \rangle) \\ = -\log p_\Psi^S(s' | \bar{s}') - \log p_{\bar{\mathcal{R}}}(r | \bar{s}, \bar{a}) - \log p_{\bar{\pi}}(\bar{a} | \bar{s}) + \log p_\phi(\bar{s}' | s') - \log p_{\text{next}}, \end{aligned}$$

where

$$\log p_{next} = \begin{cases} \log(\mathbb{E}_{a \sim \bar{\pi}(\cdot | \bar{s})} \bar{p}(\bar{s}' | \bar{s}, a)) & \text{if } \bar{\mathcal{A}} = \mathcal{A}, \text{ and} \\ \log p_{\Psi}^{\mathcal{A}}(a | \bar{s}, \bar{a}) + \log \bar{p}(\bar{s}' | \bar{s}, \bar{a}) - \log p_{\phi^{\mathcal{A}}}(\bar{a} | \bar{s}, a) & \text{otherwise} \end{cases}$$

Notice that f_{ELBO} may take negative values, so we cannot directly assign $p_{\langle s, a, r, s' \rangle}$ to f_{ELBO} since priorities are required to be strictly positive to compute the importance sampling weights. A solution is to apply a *logistic function* on $f_{\text{ELBO}} = (\langle s, a, r, s' \rangle, \langle \bar{s}, \bar{a}, \bar{s}' \rangle)$, i.e.,

$$p_{\langle s, a, r, s' \rangle} = x^* \cdot \sigma(k \cdot (f_{\text{ELBO}}(\langle s, a, r, s' \rangle, \langle \bar{s}, \bar{a}, \bar{s}' \rangle) - x_0))$$

where $x^* > 0$ is the scale, $k > 0$ is the growth rate (steepness of the curve), and x_0 is the location (midpoint) of the logistic, for some $\bar{s}, \bar{s}' \sim \phi(\cdot | \bar{s})$ and $\bar{a} \sim \phi^{\mathcal{A}}(\cdot | \bar{s}, a)$. In practice, we set x^* to the desired maximum priority and we maintain upper ($f^{\max}$) and lower ($f^{\min}$) bounds on the loss during learning. Along training steps, we tune these values according to the current loss and we set $x_0 = \frac{f^{\max} - f^{\min}}{2}$ and $k = \frac{x^*}{f^{\max} - f^{\min}}$.

2.6 Experiments

In this section, we evaluate the quality of the latent space model learned and the policy distilled via our VAE-MDP framework. This evaluation consists of: an analysis of the training of the latent space model and the benefits of our method to avoid posterior collapse, assessing the quality of the abstraction learned via PAC local losses bounds, and testing the performance of the distilled policy. This allows assessing if the latent model learned yields a sound abstraction of the state-action space that encodes the necessary information to optimize the expected return. We evaluate our method on classic OPENAI GYM environments⁶ [Brockman et al., 2016] with (i) continuous states and discrete actions (CartPole, MountainCar, and Acrobot), and (ii) where both states and actions are continuous (Pendulum and LunarLander). We distill RL policies π learned via DQN [Mnih et al., 2015] for case (i), and SAC [Haarnoja et al., 2018] for case (ii). The exact setting and labeling functions used to reproduce our results are provided in Appendix A.

Latent spaces. Since latent spaces are trained to enable formal verification, we choose $\log_2 |\bar{\mathcal{S}}|$ and $|\bar{\mathcal{A}}|$ ranging from 9 bits and 2 actions (coarser) to 16 bits and 5 actions (finer) to make them tractable for model checkers — see Budde et al. [2020] for a performance comparison of state of the art tools for a range of instances with large model size.

⁶now maintained in GYMNASIUM [Towers et al., 2023].

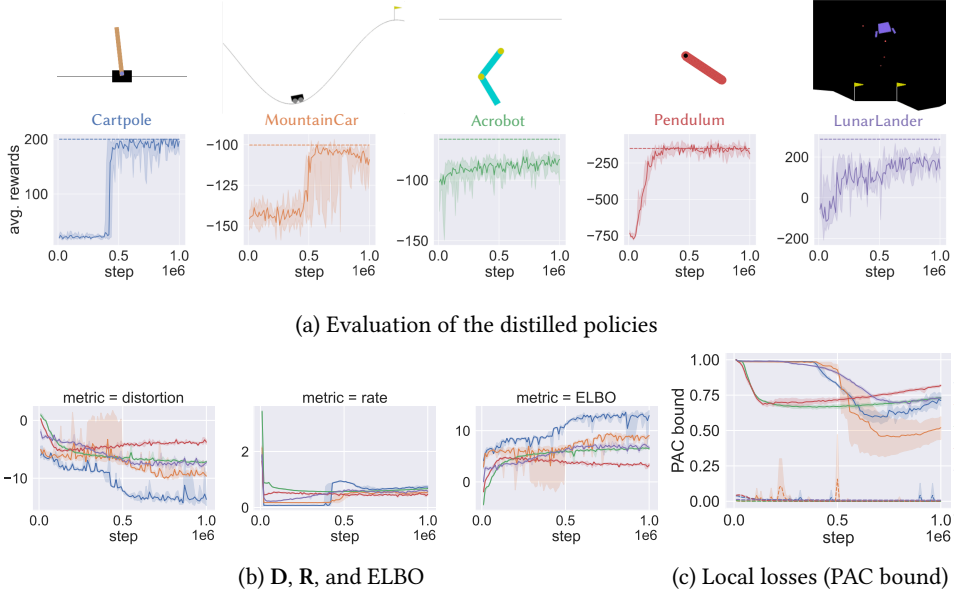


Figure 3.7: Plots reporting (3.7a) the expected episodic return (approximated by averaging over 30 episodes — the straight dashed line corresponds to the performance of the original policy); (3.7b) D , R , and $ELBO$ (approximated over a large batch sampled from the replay buffer, using discrete latent distributions and averaged using importance sampling weights); and (3.7c) PAC local losses approximation for $\epsilon = 10^{-2}$, $\delta = 5 \cdot 10^{-3}$: solid lines stand for the transition loss and dashed lines for the reward loss. For each environment, we train five different instances of our VAE with different random seeds. The solid line corresponds to the median and the shaded interval to the interquartile range

Policy distillation. The guarantees derived in Sections 1.2 and 1.3 are only valid for $\bar{\pi}$, the policy under which formal properties can be verified. We further need to evaluate if $\bar{\pi}$ achieves performance close to the original policy π that was trained to optimize the expected return in the environment. We compare the undiscounted episodic return achieved by $\bar{\pi}$ compared to π in the environment $\mathcal{M}$ (Figure 3.7a). Our framework allows learning to improve $\bar{\pi}$ along training steps and eventually achieve close performance to those of the original policy π . This illustrates that training our latent model enables distilling π into a latent policy $\bar{\pi}$.

Local losses. We report the PAC bounds on the local losses (Lemma 14), computed along training steps (Figure 3.7c). When the policy is eventually distilled (Figure 3.7a)

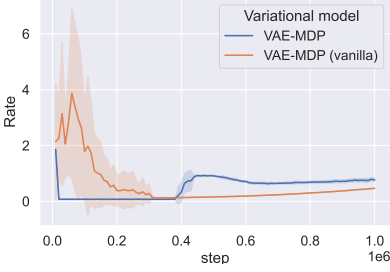
and $\bar{\pi}$ achieves performance similar to those of π , the learning curves stabilize, while maximizing the ELBO (Figure 3.7b) successfully allows lowering the local losses. In every environment where we tested our method, a low reward loss is maintained while the transition loss reaches values in the interval $[1/4, 3/4]$ for most of the instances. These values induce an upper bound on the bisimulation distance between $\mathcal{M}$ and $\bar{\mathcal{M}}$ as well as their value difference (cf. Section 1.3). We do not expect our approach to reach zero reward and transition loss though, since we pass from continuous to discrete spaces: the abstraction induced by our approach is always coarser than the original spaces, which translates in general to precision loss. This precision loss may be encoded through the discrete probability transition function. For instance, a state s which deterministically transitions to a close state s' in $\mathcal{M}$ such that $\phi(s) = \bar{s} = \phi(s')$ induces $\bar{\mathbf{P}}_{\bar{\pi}}(\bar{s} \mid \bar{s}) > 0$. In contrast to L_P , the PAC upper bound ($\hat{L}_P = 1 - 1/T \sum_{t=1}^T \bar{\mathbf{P}}(\phi(s_{t+1}) \mid \phi(s_t), \bar{a}_t)$) is sensitive to the entropy of $\bar{\mathbf{P}}$, which explains why the upper bound is sometimes particularly high. The true value of L_P is in general lower than the approximated upper bound $\hat{L}_P$.

Posterior Collapse and Annealing Scheme

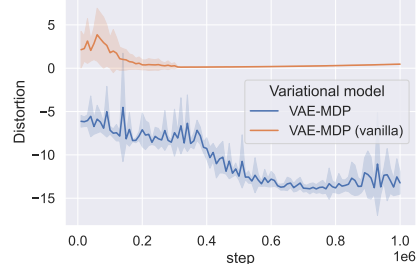
In order to allow the ELBO to be optimized in an efficient way, posterior collapse should be tackled from the very first stages of training. In fact, we found that the KL-scaling and entropy regularization annealing schemes (cf. Section 2.5) are necessary to avoid the latent space from collapsing into a single state-action pair after only a few training steps. We found the most efficient to start with an auto-encoder behavior ($\beta_0 = 0$) and a large entropy regularizer (e.g., $\alpha_0 = 10$) during the 10^4 first steps, and then anneal them via $\alpha_t = \alpha_0 \cdot (1 - \vartheta)^t$, $\beta_t = 1 - (1 - \vartheta)^t$ (with, e.g., $\vartheta = 10^{-5}$) to fully recover the original ELBO in a second training phase. We also found that prioritized experience replay further prevents posterior collapse during training (cf. Figure 3.6). Figure 3.7b shows that training ELBO this way results in a stable learning procedure that successfully minimizes $\mathbf{D}$ while preventing an auto-decoding behavior by keeping $\mathbf{R}$ away from 0.

More precisely, we illustrate the sensitivity of VAE-MDPs to the posterior collapse problem in Figure 3.8, in the CartPole environment: solely minimizing the distortion and the rate as is ($\alpha = 0$, $\beta = 1$) yields an embedding function which maps deterministically every input state to a restricted region of the latent model (cf. Figure 3.6). This indicates a form of posterior collapse: the rate quickly drops to zero (cf. Figure 3.8a), and the resulting latent representation yields no information at all.

Impact on the resulting learning procedure. The annealing scheme used to avoid every state from collapsing to the same representation possibly induces a high entropy embedding function (Figure 3.8c), which further complicates the learning of the model dynamics and the distillation in the first stage of the training process. In fact, one can observe that the entropy reaches its maximal value in Figure 3.8, which yields a fully



(a) Rate of the variational model.



(b) Distortion of the variational model.

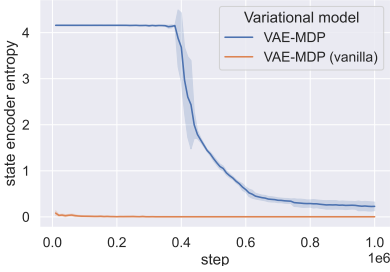
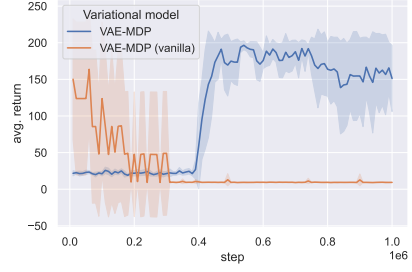
(c) Average point-wise entropy of $\phi(\cdot | s)$, for $s \in \mathcal{S}$ drawn from the interaction.(d) Performance of the resulting distilled policy $\bar{\pi}$ when deployed in the original environment.

Figure 3.8: Comparison of the VAE-MDP in the CartPole environment (i) when the distortion and the rate are solely minimized as is ($\alpha = 0, \beta = 1$, referred to as the *vanilla model*) and (ii) when we make use of annealing schemes, entropy regularization, and prioritized experience replay to avoid posterior collapse. While the former clearly fails to learn a useful latent representation, the latter does so meticulously and smoothly in two distinguishable phases: first, ϕ focuses on fairly distributing the latent space, setting up the stage for the concrete optimization occurring from step $4 \cdot 10^5$, where the entropy of ϕ is lowered, which allows to get the rate of the variational model away from zero.

random state embedding function. Recall that the VAE-MDP latent space is learned through independent Bernoulli distributions. Figure 3.8c reports values centered around 4.188 in the first training phase, which corresponds to the entropy of the state embedding function when $\phi(\cdot | s)$ is uniformly distributed over $\mathcal{S}$, for any state $s \in \mathcal{S}$.⁷ The rate (Figure 3.8a) drops to zero since the KL divergence with ϕ pulls the latent dynamics $\bar{\mathbf{P}}$ towards a fully random process (yet another form of posterior collapse), which hinders the latent space model to learn a useful representation. However, the annealing scheme increases the importance of the rate along training steps, which enables the optimization to eventually leave this local optimum (here around $4 \cdot 10^5$ training steps). This allows the learning procedure to leave the zero-rate spot, reduce the distortion (Figure 3.8b), and finally distill the original policy (Figure 3.8d).

As a result, the distillation scheme required to mitigate posterior collapse slows down the training procedure. This phenomenon is reflected in Figure 3.7: the variational model needs several steps to stabilize and set up the stage for concrete optimization.

3 Wasserstein Auto-encoded MDP

VAE-MDPs offer a valuable framework for distilling deep RL policies, which enables their verification through tractable latent space models. Although VAE-MDPs come with PAC-verifiable bisimulation bounds (Sections 1.2 and 1.3), they, unfortunately, suffer from several learning flaws.

First, training a VAE-MDP relies on variational proxies for optimizing the bisimulation bounds, which means there is no guarantee *at learning time* on the quality of the latent model derived from its optimization process (i.e., *no learning guarantee*).

Second, VAEs are known to suffer from the posterior collapse problem and VAE-MDPs are no exception (see Figure 3.8). This phenomenon results in a deterministic mapping to constrained regions of the latent space. Furthermore, analyzing the variational metrics and the entropy of the embedding function during learning revealed that most of the training process focuses on handling this phenomenon and setting up the stage for concrete distillation and abstraction, finally taking place in a second training phase. This requires extra regularizers, setting up annealing schemes and learning phases, and defining prioritized replay buffers to store the transitions. Distillation through VAE-MDPs is thus a meticulous task that may require a large step budget and tuning many hyperparameters.

Building upon *Wasserstein auto-encoders* [Tolstikhin et al., 2018] instead of VAEs, we introduce *Wasserstein auto-encoded MDPs* (WAE-MDPs), which overcome those limitations. Our WAE relies on the *optimal transport* (OT) from trace distributions

⁷ $h(\phi(\cdot | s)) = \sum_{i=0}^{\log_2 |\mathcal{S}| - |\mathbf{AP}|=6} -p_i \log p_i - (1 - p_i) \log(1 - p_i) = 4.188$, where $p_i = 1/2$ for all i .

resulting from the execution of the RL policy in the real environment to that reconstructed from the latent model operating under the distilled policy. In contrast to VAEs which rely on variational proxies, we derive a novel objective that incorporates the bisimulation bounds. Furthermore, while VAEs learn stochastic mappings to the latent space that need to be determinized at evaluation time to obtain the guarantees, our WAE has no such requirements and learns *all the necessary components to obtain the guarantees during learning*, which does not require such post-processing operations.

Those theoretical claims are reflected in our experiments: policies are distilled up to 10 times faster through WAE- than VAE-MDPs and provide better abstraction quality and performance in general, without the need for setting up annealing schemes and training phases, nor prioritized buffer and extra regularizer. Our distilled policies are able to recover the original policy performance, highlighting the representation quality offered by our new framework: the distillation is able to remove some non-robustness of the input RL policy. Finally, we formally verified *time-to-failure* properties (e.g., Pnueli 1977) to emphasize the applicability of our approach.

Optimal Transport. Along this section, we fix the MDP $\overline{\mathcal{M}} = \langle \overline{\mathcal{S}}, \overline{\mathcal{A}}, \overline{\mathcal{P}}, \overline{\mathcal{R}}, \overline{\ell}, \text{AP}, \overline{s}_I \rangle$ and $\langle \overline{\mathcal{M}}, \phi, \psi \rangle$ as a latent space model of $\mathcal{M}$ parameterized by encoding and decoding weights, respectively denoted by ι and θ . Again, our approach hinges on learning a behavioral distribution $\Psi(\cdot; \theta)$ as a latent variable model, where the latent variables reside in the latent state-action space and whose dynamics are described by the execution of a latent policy $\overline{\pi}$ in $\overline{\mathcal{M}}$. By learning Ψ , we aim at capturing the dynamics of $\mathcal{M}$ and distilling the original policy π into a latent policy $\overline{\pi}$. This can be achieved via the minimization of a suitable discrepancy between Ψ and the trace distribution of $\mathcal{M}_\pi$. In the VAE case, we considered the KL divergence, scaled by the length of the traces. This time, we consider the *optimal transport* (Definition 33) from transitions occurring along the traces of $\mathcal{M}_\pi$ to those generated by Ψ .

Similarly to the VAE case, we assume that the traces on which Ψ is trained are produced in an episodic manner, which further allows us to formalize the discrepancy in a tractable way (in the local setting).

Assumption 6: Episodic environment

We assume that the original environment $\mathcal{M}$ is episodic (Definition 16).

3.1 The Discrepancy Measure

In the following, we show that reasoning about the discrepancy between traces generated in the environment and those generated by the behavioral model in an episodic process reduces to reasoning about the discrepancy between the stationary distribution of $\mathcal{M}_\pi$

and the one of the behavioral model Ψ . Consider that $\Psi(\cdot; \theta)$ is able to generate finite traces of the form $\langle s_{0:H}, a_{0:H-1}, r_{0:H-1} \rangle$. Our goal is to find the best parameters θ which offer the most accurate reconstruction of the transitions occurring along traces issued from the execution of the policy π in the environment.

Raw transition distance. For any function f denotes by $\text{Im}(f)$ its image set. Assume that S , $\mathcal{A}$, and $\text{Im}(\mathcal{R})$ are respectively equipped with metric d_S , $d_{\mathcal{A}}$, and $d_{\mathcal{R}}$. Let us define the *raw transition metric* over *transitions* of $\mathcal{M}$, i.e., tuples of the form $\langle s, a, r, s' \rangle$, as

$$\vec{d}(\langle s_1, a_1, r_1, s'_1 \rangle, \langle s_2, a_2, r_2, s'_2 \rangle) = d_S(s_1, s_2) + d_{\mathcal{A}}(a_1, a_2) + d_{\mathcal{R}}(r_1, r_2) + d_S(s'_1, s'_2).$$

In a nutshell, $\vec{d}$ consists of the sum of the distance of all the transition components. Note that the resulting distance metric is well-defined since the sum of distances preserves the identity of indiscernible, symmetry, and triangle inequality.

Occupancy measures. To compare the transitions occurring along traces, we consider the *occupancy measure* μ_{π}^H of $\mathcal{M}_{\pi}$ (cf. Chapter 2, Section 1.4), giving the fraction of time spent in each region of the environment along traces of length $H \geq 0$. By considering the occupancy measure, we can calculate the average distance of transitions occurring along the traces of $\mathcal{M}_{\pi}$ and Ψ , weighted by their occupation probabilities.

Precisely, our goal is to compare the dynamics of the environment to those of the behavioral model, which can be achieved by comparing the occupancy measures of $\mathcal{M}_{\pi}$ and Ψ :

$$\mathcal{W}_{\vec{d}}(\mu_{\pi}^H, \Psi(H; \theta)),$$

where $\Psi(H) \in \Delta(S \times \mathcal{A} \times \text{Im}(\mathcal{R}) \times S)$ denotes the distribution over transitions occurring along traces of length H , generated from Ψ . The following Lemma enables optimizing the Wasserstein distance between the original MDP and the behavioral model when traces are drawn from episodic RL processes or infinite interactions.

Lemma 16: Ergodic interpretation of the Wasserstein metric

Assume the existence of a stationary behavioral model ξ_{Ψ} so that $\Psi(H)$ weakly converges to ξ_{Ψ} as H goes to ∞ , then

$$\lim_{H \rightarrow \infty} \mathcal{W}_{\vec{d}}(\mu_{\pi}^H, \Psi(H)) = \mathcal{W}_{\vec{d}}(\xi_{\pi}, \xi_{\Psi}).$$

Proof. The result follows then from [Villani, 2009, Corollary 6.11]: since $\mathcal{M}$ is episodic, μ_{π}^H converges almost surely to ξ_{π} as H goes to ∞ , and the Wasserstein distance between μ_{π}^H and $\Psi(H)$ converges to the Wasserstein distance between ξ_{π} and ξ_{Ψ} . $\square$

3.2 The WAE-MDP Objective

Given Lemma 16, our goal is to minimize $\mathcal{W}_d^{\rightarrow}(\xi_{\pi}, \xi_{\Psi})$. We consider ξ_{Ψ} as a latent variable model over a latent space $\mathcal{Z}$. We assume that this latent space corresponds to the one of transitions generated in the latent MDP $\overline{\mathcal{M}}$, $\mathcal{Z} = \overline{\mathcal{S}} \times \overline{\mathcal{A}} \times \overline{\mathcal{S}}$. Thus, by the law of total probability, for any (original) transition s, a, r, s' , we have

$$\xi_{\Psi}(s, a, r, s') = \int_{\overline{\mathcal{S}}} \int_{\overline{\mathcal{A}}} \int_{\overline{\mathcal{S}}} p_{\Psi}(s, a, r, s' \mid \bar{s}, \bar{a}, \bar{s}') \, d\tilde{\xi}_{\pi}(\bar{s}, \bar{a}, \bar{s}'), \quad (3.10)$$

where p_{Ψ} is the density of a transition decoder and $\tilde{\xi}_{\pi}$ denotes the stationary distribution of the latent model $\overline{\mathcal{M}}$, when the latter operates under a latent policy $\bar{\pi}$.

As proved by Bousquet et al. [2017], this formulation allows to derive a simpler form of the OT: instead of finding the optimal coupling of ξ_{π} and ξ_{Ψ} in the primal definition of $\mathcal{W}_d^{\rightarrow}(\xi_{\pi}, \xi_{\Psi})$ (Definition 33), it is sufficient to find an encoder $\vec{\phi} \in \Delta(\mathcal{Z})$ whose marginal $\Phi(\cdot) = \mathbb{E}_{s,a,s' \sim \xi_{\pi}} \vec{\phi}(\cdot \mid s, a, s')$ is identical to $\tilde{\xi}_{\pi}$. This is summarized in the following theorem, yielding a particular case of *Wasserstein auto-encoder* [Tolstikhin et al., 2018]:

Theorem 27: Wasserstein auto-encoded Markov decision process

Let ξ_{Ψ} and p_{Ψ} be a (stationary) behavioral model and the density of a transition decoder as defined in Equation 3.10, $\zeta(\cdot; \theta) : \overline{\mathcal{S}} \rightarrow \mathcal{S}$ be a (deterministic) state-wise decoder, and $\psi(\cdot; \theta) : \overline{\mathcal{S}} \times \overline{\mathcal{A}} \rightarrow \mathcal{A}$ be an action decoder.^a Assume that the transition decoder is deterministic, where $p_{\Psi}(\cdot \mid \bar{s}, \bar{a}, \bar{s}')$ consists in the Dirac impulse at $\rho(\bar{s}, \bar{a}, \bar{s}') = \langle \zeta(\bar{s}), \psi(\bar{s}, \bar{a}), \overline{\mathcal{R}}(\bar{s}, \bar{a}), \zeta(\bar{s}') \rangle$, then

$$\mathcal{W}_d^{\rightarrow}(\xi_{\pi}, \xi_{\Psi}) = \inf_{\vec{\phi} : \Phi = \tilde{\xi}_{\pi}} \mathbb{E}_{s,a,r,s' \sim \xi_{\pi}} \mathbb{E}_{\bar{s}, \bar{a}, \bar{s}' \sim \vec{\phi}(\cdot \mid s, a, s')} \vec{d}(\langle s, a, r, s' \rangle, \rho(\bar{s}, \bar{a}, \bar{s}')).$$

^aNotice that we use the same symbol for the action decoder and the action lifting function defined in Section 1, taking *original* states as input (in contrast to *latent* states). To recover $\psi : \mathcal{S} \times \overline{\mathcal{A}} \rightarrow \mathcal{A}$, one just needs to compose the decoder with the state embedding function ϕ , so that $\psi(s, \bar{a}) = \psi(\phi(s), \bar{a})$.

Henceforth, assume that the density of the encoder $\vec{\phi}$ is composed via the state embedding function $\phi(\cdot; \iota) : \mathcal{S} \rightarrow \overline{\mathcal{S}}$ and an action encoder $\phi^{\mathcal{A}}(\cdot; \iota) : \overline{\mathcal{S}} \times \mathcal{A} \rightarrow \Delta(\overline{\mathcal{A}})$ as

$$p_{\vec{\phi}}(\bar{s}, \bar{a}, \bar{s}' \mid s, a, s') = \delta_{\phi(s)}(\bar{s}) \cdot p_{\phi^{\mathcal{A}}}(\bar{a} \mid \bar{s}, a) \cdot \delta_{\phi(s')}(\bar{s}'),$$

where δ_x is the Dirac measure concentrating all its mass at the point x .

Dealing with discrete actions. We mentioned in Remark 16 that when the policy executed in $\mathcal{M}$ already produces discrete actions, learning a latent action space is, in

many cases, not necessary. Notably, when a latent policy $\bar{\pi}$ is executed in $\mathcal{M}$, it is not necessary to use the action encoder ϕ^A , since the policy already produces latent actions. We thus make the following assumptions.

Assumption 7: Selective action encoder

Let $\tilde{\mathcal{A}} \in \{\mathcal{A}, \bar{\mathcal{A}}\}$ and $\tilde{\pi} : \mathcal{S} \rightarrow \Delta(\tilde{\mathcal{A}})$ be the policy executed in $\mathcal{M}$ so that $\tilde{\mathcal{A}}$ is a (tractable) finite set. Then, consider that $\bar{\mathcal{A}} = \tilde{\mathcal{A}}$ and ϕ^A as the projection $\phi^A(\cdot, \tilde{a}) = \tilde{a}$.

Assumption 8: Trivial lifting

Assume that the action space of the original environment $\mathcal{M}$ is a (tractable) finite set. Then, we take the action lifting function ψ as the trivial projection $\psi = \phi^A$.

Concretely, the premise of Assumption 7 typically occurs when π is a latent policy *or* when $\mathcal{M}$ has already a discrete action space. In the latter case, Assumption 7 and 8 amount to setting $\bar{\mathcal{A}} = \mathcal{A}$ and ignoring the action encoding and lifting functions. Note that if a discrete action space is too large, or if the user explicitly aims for a coarser space, then the former is not considered as being tractable.

Optimizing the objective. As optimizing over encoders $\vec{\phi}$ whose marginal Φ is identical to $\tilde{\xi}_{\bar{\pi}}$ is intractable, we consider a relaxation of the problem by minimizing a discrepancy D between the marginal Φ and the latent stationary distribution $\tilde{\xi}_{\bar{\pi}}$:

$$\min_{\iota, \theta} \mathbb{E}_{s, a, r, s' \sim \tilde{\xi}_{\bar{\pi}}} \mathbb{E}_{\tilde{s}, \tilde{a}, \tilde{s}' \sim \vec{\phi}(\cdot | s, a, s'; \iota)} \vec{d}(\langle s, a, r, s' \rangle, \rho(\tilde{s}, \tilde{a}, \tilde{s}'; \theta)) + \beta \cdot D(\Phi(\cdot; \iota), \tilde{\xi}_{\bar{\pi}}), \quad (3.11)$$

where $\beta > 0$ is a scaling factor, describing the importance of penalizing the divergence between Φ and $\tilde{\xi}_{\bar{\pi}}$. Notice that similarly to the VAE case (Theorem 26), the objective function is also subdivided into a reconstruction term $\mathbb{E} [\vec{d}(\langle s, a, r, s' \rangle, \rho(\tilde{s}, \tilde{a}, \tilde{s}'))]$ and a regularizer $D(\Phi, \tilde{\xi}_{\bar{\pi}})$. However, in contrast to VAE-MDPs, the latent regularization does not aim at minimizing the divergence between the state-wise embedding and the latent transition function, but consider marginal distributions instead (namely, the marginal encoding distribution Φ and the latent stationary distribution $\tilde{\xi}_{\bar{\pi}}$).

When π operates in the original model $\mathcal{M}$, its *parallel execution* in the latent model $\bar{\mathcal{M}}$ is enabled by the action encoder ϕ^A (cf. Figure 3.9): given an original state $s \in \mathcal{S}$, π first prescribes the action $a \sim \pi(\cdot | s)$, which is then embedded in the latent space via $\tilde{a} \sim \phi^A(\cdot | \phi(s), a)$. Note that the execution of π in the latent model $\bar{\mathcal{M}}$ is dependent to its

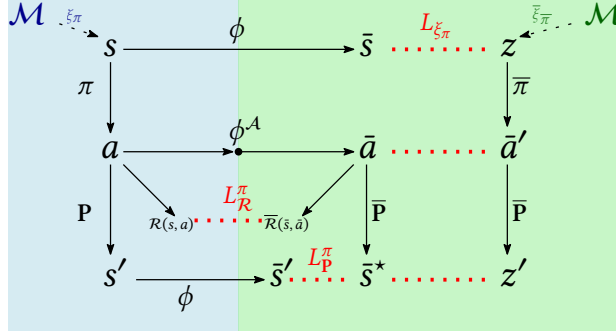


Figure 3.9: Parallel execution of the original policy π in $\mathcal{M}$ and $\bar{\mathcal{M}}$. The original and latent state-action spaces are respectively spread along the blue (left) and green (right) area. The local losses and the steady-state regularizer (for the policy π) are depicted by the dashed lines, in red.

parallel execution in $\mathcal{M}$ since the actions produced solely depend on the current state s of $\mathcal{M}$.

Assume now that the latent action space is equipped with the distance metric $d_{\bar{\mathcal{A}}}$. Then, one can naturally extend the transition metric $\vec{d}$ in $\bar{\mathcal{M}}$ as the sum of the distances in the latent space. Consider fixing the discrepancy to the Wasserstein distance (again), i.e., $D = \mathcal{W}_{\vec{d}}$. Then, the parallel execution of π allows to derive the following upper bound on the latent regularizer $\mathcal{W}_{\vec{d}}(\Phi, \xi_{\pi})$.

Lemma 17: WAE-MDP regularizer

Assume that $\bar{\mathbf{P}}$ has a density $\bar{\mathbf{p}}$. Define $\mathcal{T}_{\pi}$ as the distribution of drawing state-action pairs $s, a \sim \xi_{\pi}$ from the interaction with $\mathcal{M}$, embedding them to the latent spaces via $\bar{s} = \phi(s)$, and finally letting them transition to their successor state in the latent model via $\bar{s}' \sim \bar{\mathbf{P}}(\cdot \mid \bar{s}, \bar{a})$. We define the density of $\mathcal{T}_{\pi}$ as

$$p_{\mathcal{T}_{\pi}}(\bar{s}, \bar{a}, \bar{s}') = \mathbb{E}_{s, a \sim \xi_{\pi}} [\delta_{\phi(s)}(\bar{s}) \cdot p_{\phi^A}(\bar{a} \mid \bar{s}, a) \cdot \bar{\mathbf{p}}(\bar{s}' \mid \bar{s}, \bar{a})].$$

Then, $\mathcal{W}_{\vec{d}}(\Phi, \xi_{\pi}) \leq \mathcal{W}_{\vec{d}}(\mathcal{T}_{\pi}, \xi_{\pi}) + L_{\bar{\mathbf{P}}}^{\pi}$, where $L_{\bar{\mathbf{P}}}^{\pi}$ denotes the local transition loss *under the original policy*, precisely defined as

$$L_{\bar{\mathbf{P}}}^{\pi} = \mathbb{E}_{s, a \sim \xi_{\pi}} \mathbb{E}_{\bar{a} \sim \phi^A(\cdot \mid \phi(s), a)} \mathcal{W}_{\vec{d}}(\phi \mathbf{P}(\cdot \mid s, a), \bar{\mathbf{P}}(\cdot \mid \phi(s), \bar{a})).$$

Proof. Being a distance metric [Villani, 2009], we apply the triangle inequality on the Wasserstein, which yields

$$\mathcal{W}_d(\Phi, \bar{\xi}_\pi) \leq \mathcal{W}_d(\Phi, \mathcal{T}_\pi) + \mathcal{W}_d(\mathcal{T}_\pi, \bar{\xi}_\pi),$$

where

$$\begin{aligned} & \mathcal{W}_d(\mathcal{T}_\pi, \bar{\xi}_\pi) \\ &= \sup_{f \in \text{Lipsch}_d^1} \mathbb{E}_{s, a \sim \xi_\pi} \mathbb{E}_{\bar{s}, \bar{a} \sim \phi(\cdot | s, a)} \mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot | \bar{s}, \bar{a})} f(\bar{s}, \bar{a}, \bar{s}') - \mathbb{E}_{\bar{s} \sim \bar{\xi}_\pi} \mathbb{E}_{\bar{a} \sim \bar{\pi}(\cdot | \bar{s})} \mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot | \bar{s}, \bar{a})} f(\bar{s}, \bar{a}, \bar{s}'), \end{aligned}$$

where we write $\bar{s}, \bar{a} \sim \phi(\cdot | s, a)$ for embedding s in the latent space via $\bar{s} = \phi(s)$ and then drawing $\bar{a} \sim \phi^A(\cdot | \bar{s}, a)$; and

$$\begin{aligned} & \mathcal{W}_d(\Phi, \mathcal{T}_\pi) \\ &= \sup_{f \in \text{Lipsch}_d^1} \mathbb{E}_{s, a, s' \sim \xi_\pi} \mathbb{E}_{\bar{s}, \bar{a}, \bar{s}' \sim \phi(\cdot | s, a, s')} f(\bar{s}, \bar{a}, \bar{s}') - \mathbb{E}_{s, a \sim \xi_\pi} \mathbb{E}_{\bar{s}, \bar{a} \sim \phi(\cdot | s, a)} \mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}_\theta(\cdot | \bar{s}, \bar{a})} f(\bar{s}, \bar{a}, \bar{s}') \end{aligned} \quad (3.12)$$

$$\leq \mathbb{E}_{s, a \sim \xi_\pi} \mathbb{E}_{\bar{s}, \bar{a} \sim \phi(\cdot | s, a)} \sup_{f \in \text{Lipsch}_d^1} \mathbb{E}_{\bar{s}' \sim \mathbf{P}(\cdot | s, a)} f(\bar{s}, \bar{a}, \phi(s')) - \mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot | \bar{s}, \bar{a})} f(\bar{s}, \bar{a}, \bar{s}') \quad (3.13)$$

$$\begin{aligned} &= \mathbb{E}_{s, a \sim \xi_\pi} \mathbb{E}_{\bar{a} \sim \phi^A(\cdot | \phi(s), a)} \sup_{f \in \text{Lipsch}_{d_{\bar{\mathcal{S}}}}^1} \mathbb{E}_{\bar{s}' \sim \phi \mathbf{P}(\cdot | s, a)} f(\bar{s}', \bar{a}) - \mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot | \phi(s), \bar{a})} f(\bar{s}', \bar{a}) \quad (3.14) \\ &= \mathbb{E}_{s, a \sim \xi_\pi} \mathbb{E}_{\bar{a} \sim \phi^A(\cdot | \phi(s), a)} \mathcal{W}_d(\phi \mathbf{P}(\cdot | s, a), \bar{\mathbf{P}}(\cdot | \phi(s), \bar{a})). \end{aligned}$$

We pass from Equation 3.12 to Equation 3.13 by applying Lemma 10. To see how we pass from Equation 3.13 to Equation 3.14, notice that

$$\begin{aligned} \text{Lipsch}_d^1 &= \left\{ f : f(\bar{s}_1, \bar{a}_1, \bar{s}'_1) - f(\bar{s}_2, \bar{a}_2, \bar{s}'_2) \leq \vec{d}(\langle \bar{s}_1, \bar{a}_1, \bar{s}'_1 \rangle, \langle \bar{s}_2, \bar{a}_2, \bar{s}'_2 \rangle) \right\} \\ &= \left\{ f : f(\bar{s}_1, \bar{a}_1, \bar{s}'_1) - f(\bar{s}_2, \bar{a}_2, \bar{s}'_2) \leq d_{\bar{\mathcal{S}}}(\bar{s}_1, \bar{s}_2) + d_{\bar{\mathcal{A}}}(\bar{a}_1, \bar{a}_2) + d_{\bar{\mathcal{S}}}(\bar{s}'_1, \bar{s}'_2) \right\} \end{aligned}$$

Observe now that $\bar{s}$ and $\bar{a}$ are fixed in the supremum of Equation 3.13: all functions f considered and taken from Lipsch_d^1 are of the form $f(\bar{s}, \bar{a}, \cdot)$. It is thus sufficient to consider the supremum over functions from the following subset of Lipsch_d^1 :

$$\begin{aligned} & \left\{ f : f(\bar{s}, \bar{a}, \bar{s}'_1) - f(\bar{s}, \bar{a}, \bar{s}'_2) \leq d_{\bar{\mathcal{S}}}(\bar{s}, \bar{s}) + d_{\bar{\mathcal{A}}}(\bar{a}, \bar{a}) + d_{\bar{\mathcal{S}}}(\bar{s}'_1, \bar{s}'_2) \right\} \\ & \quad \text{(for } \bar{s}, \bar{a} \text{ drawn from } \phi) \\ &= \left\{ f : f(\bar{s}, \bar{a}, \bar{s}'_1) - f(\bar{s}, \bar{a}, \bar{s}'_2) \leq d_{\bar{\mathcal{S}}}(\bar{s}'_1, \bar{s}'_2) \right\} \\ &= \left\{ f : f(\bar{s}'_1) - f(\bar{s}'_2) \leq d_{\bar{\mathcal{S}}}(\bar{s}'_1, \bar{s}'_2) \right\} \\ &= \text{Lipsch}_{d_{\bar{\mathcal{S}}}}^1. \end{aligned}$$

This concludes the proof. $\square$

We thus define the WAE-MDP objective as an upper bound on Equation 3.11:

WAE-MDP objective

$$\min_{\iota, \theta} \mathbb{E}_{\substack{s, a, s' \sim \xi_\pi \\ \bar{s}, \bar{a}, \bar{s}' \sim \phi(\cdot | s, a, s')}} \left[d_S(s, \zeta(\bar{s})) + d_A(a, \psi(\bar{s}, \bar{a})) + d_S(s', \zeta(\bar{s}')) \right] + L_R^\pi + \beta \cdot (L_P^\pi + L_{\xi_\pi}^\pi),$$

where $L_R^\pi = \mathbb{E}_{s, a \sim \xi_\pi} \mathbb{E}_{\bar{a} \sim \phi^A(\cdot | \phi(s), a)} d_R(\mathcal{R}(s, a), \bar{\mathcal{R}}(\phi(s), \bar{a}))$ and $L_{\xi_\pi}^\pi = \mathcal{W}_d^2(\mathcal{T}_\pi, \xi_\pi)$. We respectively refer to L_P^π as the *transition regularizer* and $L_{\xi_\pi}^\pi$ as the *steady-state regularizer*.

The transition regularizer allows learning the latent dynamics. In turn, the steady-state regularizer allows to quantify the distance between the stationary distribution respectively induced by π in $\mathcal{M}$ and the one induced by $\bar{\pi}$ in $\bar{\mathcal{M}}$, further enabling the distillation. Note that L_R^π and L_P^π are not sufficient to ensure the bisimulation bounds (Section 1.2): running π in $\bar{\mathcal{M}}$ depends on the parallel execution of π in the original model, which does not permit its (conventional) verification. Breaking this dependency is enabled by learning to distill π into $\bar{\pi}$ through $L_{\xi_\pi}^\pi$: minimizing $L_{\xi_\pi}^\pi$ allows to make ξ_π and $\xi_{\bar{\pi}}$ closer together, further bridging the gap between the behaviors respectively induced by π and $\bar{\pi}$. When $\bar{\pi}$ is executed, one recovers the local losses L_R and L_P along with the associated bisimulation bounds in the objective function of the WAE-MDP.

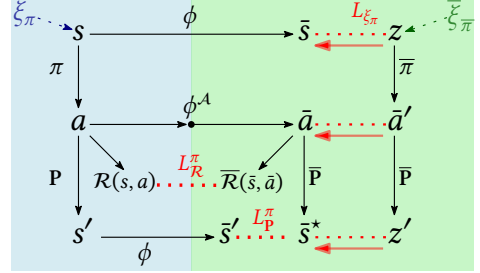


Figure 3.10: Distillation of π into $\bar{\pi}$ by minimizing $L_{\xi_\pi}^\pi$.

Theorem 28: WAE-MDP objective (under a latent policy)

Assume that traces are generated by running a latent policy $\bar{\pi}$ in the original environment and let d_R be the usual Euclidean distance, then the WAE-MDP objective reduces to

$$\min_{\iota, \theta} \mathbb{E}_{s, a, s' \sim \xi_\pi} \left[d_S(s, \zeta(\phi(s))) + d_S(s', \zeta(\phi(s'))) \right] + L_R + \beta \cdot (L_P + L_{\xi_\pi}^\pi).$$

Proof. Since the latent policy $\bar{\pi}$ generates latent actions, note that Assumption 7 holds, which means:

$$\begin{aligned} L_{\mathbf{P}}^{\bar{\pi}} &= \mathbb{E}_{s, \bar{a} \sim \xi_{\bar{\pi}}} \mathbb{E}_{\bar{a}' \sim \phi^{\mathcal{A}}(\cdot | \phi(s), \bar{a})} \mathcal{W}_{\bar{a}}(\phi \mathbf{P}(\cdot | s, \bar{a}), \bar{\mathbf{P}}(\cdot | \phi(s), \bar{a}')) \\ &= \mathbb{E}_{s, \bar{a} \sim \xi_{\bar{\pi}}} \mathcal{W}_{\bar{a}}(\phi \mathbf{P}(\cdot | s, \bar{a}), \bar{\mathbf{P}}(\cdot | \phi(s), \bar{a})) \\ &= L_{\mathbf{P}}. \end{aligned}$$

The same reasoning applies to the reward loss $L_{\mathcal{R}}$ since $d_{\mathcal{R}}$ is the Euclidean distance (or even the L_1 distance since rewards are scalar values):

$$L_{\mathcal{R}}^{\bar{\pi}} = \mathbb{E}_{s, \bar{a} \sim \xi_{\bar{\pi}}} \mathbb{E}_{\bar{a}' \sim \phi^{\mathcal{A}}(\cdot | \phi(s), \bar{a})} d_{\mathcal{R}}(\mathcal{R}(s, \bar{a}), \bar{\mathcal{R}}(\phi(s), \bar{a}')) = \mathbb{E}_{s, \bar{a} \sim \xi_{\bar{\pi}}} |\mathcal{R}(s, \bar{a}) - \bar{\mathcal{R}}(\phi(s), \bar{a})| = L_{\mathcal{R}}.$$

Then, to understand why the action decoding term vanishes, we distinguish two cases: (i) the case where the original and latent models share the same discrete action space, i.e., $\mathcal{A} = \bar{\mathcal{A}}$, and (ii) the case where the two have a different action space (e.g., when the original action space is continuous), i.e., $\mathcal{A} \neq \bar{\mathcal{A}}$.

For case (i), the result naturally follows from Assumption 8: the action lifting function is merely a projection and can simply be ignored.

In case (ii), only Assumption 7 holds, meaning the action encoder is ignored, but not the action lifting function. Given $s \sim \xi_{\bar{\pi}}$, recall that executing $\bar{\pi}$ in $\mathcal{M}$ amounts to embedding the produced latent actions $\bar{a} \sim \bar{\pi}(\cdot | \phi(s))$ back to the original environment via $a = \psi(\phi(s), \bar{a})$, resulting in the next (original) state

$$s' \sim \mathbf{P}(\cdot | s, \psi(\phi(s), \bar{a})) \equiv s' \sim \mathbf{P}(\cdot | s, \bar{a}) \quad (\text{cf. Section 1 and Figure 3.3}).$$

Therefore, given $r = \mathcal{R}(s, a)$, the projection of $\vec{d}(\langle s, a, r, s' \rangle, \rho(\phi(s), \bar{a}, \phi(s')))$ on the action space $\mathcal{A}$ is $d_{\mathcal{A}}(a, \psi(\phi(s), \bar{a})) = 0$. $\square$

Remark 22: Beyond distillation

Theorem 28 also suggests that when one desires to use the WAE-MDP as a general latent space learner to leverage the state representation induced by ϕ , the objective directly minimizes the local losses $L_{\mathcal{R}}$ and $L_{\mathbf{P}}$. The minimization of the local losses further ensures that the representation is well suited for optimizing an RL policy: by Theorem 24, states sharing the same embedding are bisimilarly close and are thus guaranteed to yield a close expected return.

Interestingly, encoding the actions is not necessary in that case (Assumption 7) and the action decoding loss vanishes. This means that when our latent model is

used in the context of RL by leveraging the latent representation as input of the policy, the action lifting function is purely learned via the policy optimization — through the lifting of the policy, i.e., $\bar{a} \sim \bar{\pi}(\cdot \mid \phi(s))$ and $a = \psi(s, \bar{a})$ (see Section 1 and Figure 3.3).

These observations are beyond the scope of this chapter but will be useful in the subsequent chapters of this thesis.

3.3 Optimizing the Regularizers

We enable the optimization of the regularizers by considering the Kantorovich-Rubinstein dual formalization of the Wasserstein (Theorem 17): we introduce two networks parameterized by some weight vector ϑ , namely $\Gamma_\xi(\cdot; \vartheta)$ and $\Gamma_P(\cdot; \vartheta)$, that are constrained to be 1-Lipschitz and trained to attain the supremum of the Wasserstein dual:

$$\begin{aligned} L_{\xi_\pi}^\pi &= \max_{\vartheta : \Gamma_\xi(\cdot; \vartheta) \in \text{LipSch}_d^1} \mathbb{E}_{s, a \sim \xi_\pi} \mathbb{E}_{\bar{a} \sim \phi^A(\cdot \mid \phi(s), a)} \mathbb{E}_{\bar{s}^* \sim \bar{P}(\cdot \mid \phi(s), \bar{a})} \Gamma_\xi(\phi(s), \bar{a}, \bar{s}^*) - \mathbb{E}_{z, \bar{a}', z' \sim \xi_\pi} \Gamma_\xi(z, \bar{a}', z') \\ L_P^\pi &= \mathbb{E}_{s, a \sim \xi_\pi} \mathbb{E}_{\bar{a} \sim \phi^A(\cdot \mid \phi(s), a)} \max_{\vartheta : \Gamma_P(\cdot; \vartheta) \in \text{LipSch}_d^1} \mathbb{E}_{\bar{s}' \sim \phi P(\cdot \mid s, a)} \Gamma_P(\bar{s}') - \mathbb{E}_{\bar{s}' \sim \bar{P}(\cdot \mid \phi(s), \bar{a})} \Gamma_P(\bar{s}'). \end{aligned}$$

Notice that under this form, optimizing L_P^π is intractable due to the expectation over the maximum. Intuitively, this involves maximizing $\mathbb{E}_{\phi P} \Gamma_P(\bar{s}') - \mathbb{E}_{\bar{P}} \Gamma_P(\bar{s}')$ for *each sample* s, a drawn from the interaction, which is not affordable. Furthermore, $\mathbb{E}_{\phi P} \Gamma_P(\bar{s}')$ is computed over samples $s' \sim P(\cdot \mid s, a)$ (which are then mapped to the latent space via $\phi(s')$), and we need to have access to the transition function $P(\cdot \mid s, a)$ to estimate this expectation. To perform gradient updates based on Monte Carlo estimations, we need to extract this maximization out of the expectation. The following Lemma allows us to rewrite L_P^π to make the optimization tractable.

Lemma 18: Expected Wasserstein metric over conditional distributions

Let $\mathcal{X}, \mathcal{Y}$ be two measurable sets, $\xi \in \Delta(\mathcal{X})$, $P : \mathcal{X} \rightarrow \Delta(\mathcal{Y})$, $Q : \mathcal{X} \rightarrow \Delta(\mathcal{Y})$, and $d : \mathcal{Y} \times \mathcal{Y} \rightarrow [0, +\infty)$ be a metric on $\mathcal{Y}$. Then,

$$\mathbb{E}_{x \sim \xi} \mathcal{W}_d(P(\cdot \mid x), Q(\cdot \mid x)) = \sup_{\Gamma : \mathcal{X} \rightarrow \text{LipSch}_d^1} \mathbb{E}_{x \sim \xi} \left[\mathbb{E}_{y_1 \sim P(\cdot \mid x)} \Gamma(x)(y_1) - \mathbb{E}_{y_2 \sim Q(\cdot \mid x)} \Gamma(x)(y_2) \right]$$

Proof. Our goal is to show that

$$\mathbb{E}_{x \sim \xi} \left[\sup_{f \in \text{LipSch}_d^1} \mathbb{E}_{y_1 \sim P(\cdot \mid x)} f(y_1) - \mathbb{E}_{y_2 \sim Q(\cdot \mid x)} f(y_2) \right] \quad (3.15)$$

$$= \sup_{\Gamma : \mathcal{X} \rightarrow \text{LipSch}_d^1} \mathbb{E}_{x \sim \xi} \left[\mathbb{E}_{y_1 \sim P(\cdot|x)} \Gamma(x)(y_1) - \mathbb{E}_{y_2 \sim Q(\cdot|x)} \Gamma(x)(y_2) \right] \quad (3.16)$$

We start with (3.15) $\leq$ (3.16). Construct $\Gamma^* : \mathcal{X} \rightarrow \text{LipSch}_d^1$ by setting for all $x \in \mathcal{X}$

$$\Gamma^*(x) = \arg \sup_{f \in \text{LipSch}_d^1} \mathbb{E}_{y_1 \sim P(\cdot|x)} f(y_1) - \mathbb{E}_{y_2 \sim Q(\cdot|x)} f(y_2).$$

This gives us

$$\begin{aligned} & \mathbb{E}_{x \sim \xi} \left[\sup_{f \in \text{LipSch}_d^1} \mathbb{E}_{y_1 \sim P(\cdot|x)} f(y_1) - \mathbb{E}_{y_2 \sim Q(\cdot|x)} f(y_2) \right] \\ &= \mathbb{E}_{x \sim \xi} \left[\mathbb{E}_{y_1 \sim P(\cdot|x)} \Gamma^*(x)(y_1) - \mathbb{E}_{y_2 \sim Q(\cdot|x)} \Gamma^*(x)(y_2) \right] \end{aligned}$$

Since $\Gamma^* \in \{\Gamma \mid \Gamma : \mathcal{X} \rightarrow \text{LipSch}_d^1\}$,

$$\begin{aligned} & \mathbb{E}_{x \sim \xi} \left[\mathbb{E}_{y_1 \sim P(\cdot|x)} \Gamma^*(x)(y_1) - \mathbb{E}_{y_2 \sim Q(\cdot|x)} \Gamma^*(x)(y_2) \right] \\ & \leq \sup_{\Gamma : \mathcal{X} \rightarrow \text{LipSch}_d^1} \mathbb{E}_{x \sim \xi} \left[\mathbb{E}_{y_1 \sim P(\cdot|x)} \Gamma(x)(y_1) - \mathbb{E}_{y_2 \sim Q(\cdot|x)} \Gamma(x)(y_2) \right]. \end{aligned}$$

It remains to show that (3.15) $\geq$ (3.16). Take

$$\Gamma^* = \arg \sup_{\Gamma : \mathcal{X} \rightarrow \text{LipSch}_d^1} \mathbb{E}_{x \sim \xi} \left[\mathbb{E}_{y_1 \sim P(\cdot|x)} \Gamma(x)(y_1) - \mathbb{E}_{y_2 \sim Q(\cdot|x)} \Gamma(x)(y_2) \right].$$

Then, for all $x \in \mathcal{X}$, we have $\Gamma^*(x) \in \text{LipSch}_d^1$ which means:

$$\begin{aligned} & \mathbb{E}_{y_1 \sim P(\cdot|x)} \Gamma^*(x)(y_1) - \mathbb{E}_{y_2 \sim Q(\cdot|x)} \Gamma^*(x)(y_2) \\ & \leq \sup_{f \in \text{LipSch}_d^1} \mathbb{E}_{y_1 \sim P(\cdot|x)} f(y_1) - \mathbb{E}_{y_2 \sim Q(\cdot|x)} f(y_2) \end{aligned}$$

This finally yields

$$\begin{aligned} & \mathbb{E}_{x \sim \xi} \left[\mathbb{E}_{y_1 \sim P(\cdot|x)} \Gamma^*(x)(y_1) - \mathbb{E}_{y_2 \sim Q(\cdot|x)} \Gamma^*(x)(y_2) \right] \\ & \leq \mathbb{E}_{x \sim \xi} \left[\sup_{f \in \text{LipSch}_d^1} \mathbb{E}_{y_1 \sim P(\cdot|x)} f(y_1) - \mathbb{E}_{y_2 \sim Q(\cdot|x)} f(y_2) \right]. \end{aligned}$$

□

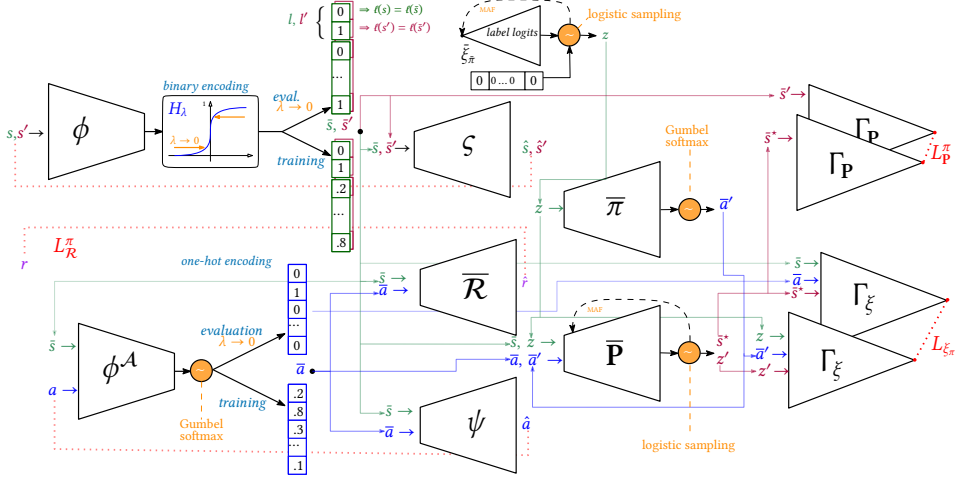


Figure 3.11: WAE-MDP architecture. Distances are depicted by red dotted lines.

Corollary 4: Transition regularizer

Let ξ_π be a stationary distribution of $\mathcal{M}_\pi$ and $\mathcal{X} = S \times \mathcal{A} \times \bar{S} \times \bar{\mathcal{A}}$, then

$$L_P^\pi = \sup_{\Gamma_P : \mathcal{X} \rightarrow \text{Lipsch}_2^1} \mathbb{E}_{s, a, s' \sim \xi_\pi} \mathbb{E}_{\tilde{s}, \tilde{a} \sim \phi(\cdot | s, a)} \left[\Gamma_P(s, a, \tilde{s}, \tilde{a})(\phi(s')) - \mathbb{E}_{\tilde{s}' \sim \bar{P}(\cdot | \tilde{s}, a)} \Gamma_P(s, a, \tilde{s}, \tilde{a})(\tilde{s}') \right].$$

The networks are constrained via the gradient penalty approach of Gulrajani et al. [2017], leveraging that any differentiable function is 1-Lipschitz if and only if it has gradients with norm at most 1 everywhere. Then, we add a gradient penalty term to the objective function to encourage the norm of the gradients to be close to 1 and enforce the Lipschitz constraint. It is worth mentioning that the applicability of the gradient penalty trick depends on the latent metric $\tilde{a}$. We dwell on the latent metric that we use in our latent model in Section 3.5, where we further assess the applicability of the gradient penalty approach.

The final learning process is presented in Algorithm 6.

3.4 Discrete Latent Model

To enable the verification of latent models supported by the bisimulation guarantees of Sections 1.2 and 1.3, we focus on the special case of *discrete* latent space models. As in VAE-MDPs, our approach relies on continuous relaxations of discrete random variables,

Algorithm 6: Wasserstein Auto-Encoded MDP

Input: batch size N , max. step T , no. of regularizer updates n_{reg} between each reconstruction, penalty coefficient $\alpha > 0$

for $t = 1$ **to** T **do**

for $i = 1$ **to** N **do**

 Sample a transition s_i, a_i, r_i, s'_i from the original environment via ξ_π

 Embed the transition into the latent space: $\bar{s}_i, \bar{a}_i, \bar{s}'_i \sim \vec{\phi}(\cdot \mid s_i, a_i, s'_i)$

 Make the latent space model transition to the next latent state:

$\bar{s}_i^* \sim \bar{\mathbf{P}}(\cdot \mid \bar{s}_i, \bar{a}_i)$

 Sample a latent transition from $\tilde{\xi}_\pi$:

$z_i \sim \tilde{\xi}_\pi$, $\bar{a}'_i \sim \bar{\pi}(\cdot \mid z_i)$, and $z'_i \sim \bar{\mathbf{P}}(\cdot \mid z_i, \bar{a}'_i)$

$\mathcal{W} \leftarrow \sum_{i=1}^N \Gamma_\xi(\bar{s}_i, \bar{a}_i, \bar{s}_i^*) - \Gamma_\xi(z_i, \bar{a}'_i, z'_i) + \Gamma_{\mathbf{P}}(s_i, a_i, \bar{s}_i, \bar{a}_i, \bar{s}'_i) - \Gamma_{\mathbf{P}}(s_i, a_i, \bar{s}_i, \bar{a}_i, \bar{s}_i^*)$

$P \leftarrow \sum_{i=1}^N \text{GP}(\Gamma_\xi, \langle \bar{s}_i, \bar{a}_i, \bar{s}_i^* \rangle, \langle z_i, \bar{a}'_i, z'_i \rangle) + \text{GP}(\mathbf{x} \mapsto \Gamma_{\mathbf{P}}(s_i, a_i, \bar{s}_i, \bar{a}_i, \mathbf{x}), \bar{s}'_i, \bar{s}_i^*)$

 Update the Lipschitz network parameters ϑ by ascending $1/N \cdot (\beta \mathcal{W} - \alpha P)$

if $t \bmod n_{\text{reg}} = 0$ **then**

$\mathcal{L} \leftarrow \sum_{i=1}^N d_S(s_i, \varsigma(\bar{s}_i)) + d_{\mathcal{A}}(a_i, \psi(\bar{s}_i, \bar{a}_i)) + d_{\mathcal{R}}(r_i, \bar{\mathcal{R}}(\bar{s}_i, \bar{a}_i)) + d_S(s'_i, \varsigma(\bar{s}'_i))$

 Update the latent space model parameters $\langle \iota, \theta \rangle$ by descending

$1/N \cdot (\mathcal{L} + \beta \mathcal{W})$

function $\text{GP}(\Gamma, \mathbf{x}, \mathbf{y})$

$\triangleright$ **Gradient penalty** for $\Gamma : \mathbb{R}^n \rightarrow \mathbb{R}$ and $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$

$\epsilon \sim U(0, 1)$

$\triangleright$ random noise

$\tilde{\mathbf{x}} \leftarrow \epsilon \mathbf{x} + (1 - \epsilon) \mathbf{y}$

$\triangleright$ straight lines between $\mathbf{x}$ and $\mathbf{y}$

return $(\|\nabla_{\tilde{\mathbf{x}}} \Gamma(\tilde{\mathbf{x}})\| - 1)^2$

regulated by some temperature parameter λ : discrete random variables are retrieved as $\lambda \rightarrow 0$, which amounts to applying a rounding operator (cf. Section 2.4 for more details on continuous relaxation of discrete distributions). During training, we use temperature-controlled relaxations to differentiate the objective and let the gradient flow through the network. When we deploy the latent policy in the environment and formally check the latent model, the zero-temperature limit is used. An overview of the approach and the WAE-MDP architecture is depicted in Figure 3.11.

Remark 23: Continuous relaxations of discrete distributions

In contrast to VAEs which require (strictly) stochastic^a and continuous densities to optimize the ELBO, the WAE-MDP objective does not have such requirements. Then, alternatively to using continuous relaxation of discrete distributions, one could use the *straight-through gradients estimator* [Bengio et al., 2013], as used by van den Oord et al. [2017]; Fajtl et al. [2020] and Hafner et al. [2021]. This consists of using discrete variables and non-differentiable functions in the forward pass of the input through the neural networks, while continuous variables and surrogate functions are used in the backward pass, i.e., during the backpropagation of the gradients. This yields low-variance, but biased gradients [Hafner et al., 2021; Shekhovtsov, 2021]. In contrast, continuous relaxations allow interpolating between these phenomena: a higher temperature produces low-variance but biased gradients while lowering the temperature to zero increases the variance but ends up producing unbiased gradients. For this reason, we stick to continuous relaxations for modeling our distributions. When the objective is optimized, relaxed distributions are used ($\lambda > 0$), and their discrete counterparts are then used afterward (zero-temperature limit).

^aprecisely, the support of the target distributions should be included in the support of the measure induced by the learned densities, otherwise the computation may produce $\log(0)$, which is undefined.

State encoder. We work with a *binary representation* of the latent states. First, this induces compact networks, able to deal with a large discrete space via a tractable number of parameter variables. But most importantly, this ensures that Assumption 5 is satisfied: let $n = \log_2 |\bar{\mathcal{S}}|$, as for VAE-MDPs, we reserve $|\mathbf{AP}|$ bits in $\bar{\mathcal{S}}$ and each time $s \in \mathcal{S}$ is passed to ϕ , $n - |\mathbf{AP}|$ bits are produced and concatenated with $\ell(s)$, ensuring a perfect reconstruction of the labels (details in Section 2.4). To produce Bernoulli variables, ϕ deterministically maps s to a latent code $\mathbf{z}$, passed to the Heaviside $H(\mathbf{z}) = \mathbb{1}\{\mathbf{z} > 0\}$. We train ϕ by using the smooth approximation $H_\lambda(\mathbf{z}) = \sigma(2\mathbf{z}/\lambda)$, satisfying $H = \lim_{\lambda \rightarrow 0} H_\lambda$.

Latent distributions. Concerning the other latent distributions, we use the Gumbel softmax trick to sample from categorical distributions (ϕ^A and $\bar{\pi}$), further producing

relaxed, one-hot latent actions. For distributions over binary vectors (modeling $\bar{\mathbf{P}}$ and $\xi_{\bar{\pi}}$), we see each bit as a relaxed Bernoulli with logit α , which is retrieved by drawing a logistic random variable located in α/λ and scaled to $1/\lambda$, where a sigmoid is applied in downstream. Additional details on the Gumbel softmax trick and the relaxation of discrete distributions are provided in Section 2.4. We stress that using independent Bernoulli distributions (as used in VAE-MDPs and by, e.g., Corneil et al., 2018) may be insufficient: independent Bernoullis are too restrictive in general, which prevents from learning sound transition dynamics, as illustrated in Example 17.

Example 17: Independent Bernoullis

Let $\bar{\mathcal{M}}$ be the discrete MC of Fig. 3.12 (the labels are drawn next to their state). In one-hot, $\mathbf{AP} = \{\text{goal} : \langle 1, 0 \rangle, \text{unsafe} : \langle 0, 1 \rangle\}$. We assume that 3 bits are used for the (binary) state space, with $\bar{\mathcal{S}} = \{\bar{s}_0 : \langle 0, 0, 0 \rangle, \bar{s}_1 : \langle 1, 0, 0 \rangle, \bar{s}_2 : \langle 0, 1, 0 \rangle, \bar{s}_3 : \langle 0, 1, 1 \rangle\}$ (the two first bits are reserved for the labels). Considering each bit as being independent is not sufficient to learn the transition function: the optimal estimation $P(\cdot | \bar{s}_0)$ is in that case represented by the independent probability vector $\mathbf{b} = \langle 1/2, 1/2, 1/4 \rangle$, giving the probability to go from $\bar{s}_0$ to each bit *independently*. This yields a poor estimation of the actual transition function: $P(\bar{s}_0 | \bar{s}_0) = (1 - \mathbf{b}_1) \cdot (1 - \mathbf{b}_2) \cdot (1 - \mathbf{b}_3) = P(\bar{s}_1 | \bar{s}_0) = \mathbf{b}_1 \cdot (1 - \mathbf{b}_2) \cdot (1 - \mathbf{b}_3) = P(\bar{s}_2 | \bar{s}_0) = (1 - \mathbf{b}_1) \cdot \mathbf{b}_2 \cdot (1 - \mathbf{b}_3) = 3/16$; and $P(\bar{s}_3 | \bar{s}_0) = (1 - \mathbf{b}_1) \cdot \mathbf{b}_2 \cdot \mathbf{b}_3 = 1/16$.

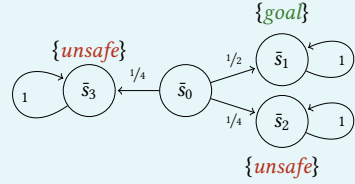


Figure 3.12: Markov Chain with four states.

Autoregressive relaxed Bernoullis. To learn a *multivariate* Bernoulli distribution, we use a *masked autoregressive flow* (MAF, Papamakarios et al., 2017) to model $\bar{\mathbf{P}}$.

Definition 40: Autoregressive density

Let $\mathcal{X}$ be a complete and separable metric space and $P \in \Delta(\mathcal{X}^n)$ be a multivariate distribution over $n > 1$ dimensions with density p . By the chain rule, one can rewrite $p(\mathbf{x})$ as $p(\mathbf{x}) = p(x_1) \cdot \prod_{i=2}^n p(x_i | x_1, \dots, x_{i-1})$ for all $\mathbf{x} \in \mathcal{X}^n$, where x_i denotes the element on the i^{th} dimension of vector $\mathbf{x}$.

We say that the density p is *autoregressive* if there is a function $f : \mathcal{X}^n \rightarrow \mathcal{Y}^n$ that generates the parameters of a (univariate) density function p' over $\mathcal{X}$ so that $f(\mathbf{x}) = \mathbf{z}$ implies $p'(x_i; z_i) = p(x_i | x_1, \dots, x_{i-1})$ for all $i \in \{2, \dots, n\}$ and $p'(x_1; z_1) = p(x_1)$.

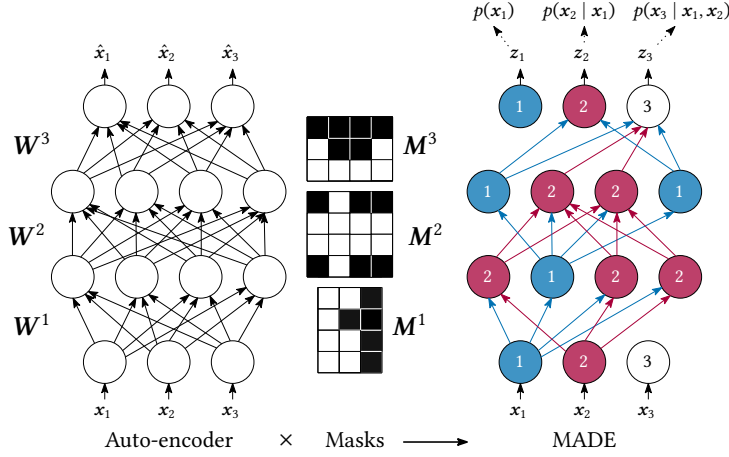


Figure 3.13: Vanilla auto-encoder (left), binary masks (center), and MADE (right) [Germain et al., 2015]. The networks process the inputs (bottom) to the outputs (top). The units are partitioned into 2 groups via a coloring function (blue for 1, red for 2).

In particular, a *masked auto-encoder for density estimation* (MADE, Germain et al., 2015) is a special kind of feedforward network that models the function f via an auto-encoder architecture. Intuitively, the input layer is connected to the output layer through connections that are masked to ensure the autoregressive property.

Definition 41: Masked auto-encoder for density estimation

A *masked auto-encoder for density estimation* (MADE) is a feed-forward network $f : \mathcal{X}^n \rightarrow \mathcal{Y}^n$ (cf. Definition 15), where $\mathcal{Y}$ is the parameter space of distributions over $\mathcal{X} \subseteq \mathbb{R}$. The network f is parameterized by an activation function σ_l , weights $\mathbf{W}^l \in \mathbb{R}^{n_{l-1} \times n_l}$ and biases $\mathbf{b}^l \in \mathbb{R}^{n_l}$, as well as *masks* $\mathbf{M}^l \in \{0, 1\}^{n_{l-1} \times n_l}$ for each layer $l \in \{1, \dots, L-1\}$, where n_l is the number of neurons in layer l . Then, the output of the j^{th} hidden unit z_j^l in layer l is given by $z_j^l = \sum_{i=1}^{n_{l-1}} \sigma_l(z_i^{l-1} \cdot \mathbf{M}_{i,j}^l \cdot \mathbf{W}_{i,j}^l + \mathbf{b}_j^l)$.

To ensure the autoregressive property, a *color* $c_l : \{1, \dots, n_l\} \rightarrow \{1, \dots, n-1\}$ is assigned to each unit in layer $l < L$, with $c_0(i) = i$ for all $i \in \{1, \dots, n-1\}$. This allows partitioning the neurons into $n-1$ groups, where each unit with color k indicates that it is connected to inputs $x_1, \dots, x_k$. The last input x_n is not colored as none of the units in the output layer should depend on x_n . Precisely, $\mathbf{M}_{i,j}^l = \mathbb{1}_{\{c_{l-1}(i) \leq c_l(j)\}}$ for $1 \leq l < L-1$. This allows to flow the information from inputs $x_1, \dots, x_{c_l(j)}$ to the j^{th} unit of layer l . For the output layer, we set $\mathbf{M}_{i,j}^{L-1} = \mathbb{1}_{\{c_{L-2}(i) < j\}}$, which allows to condition the j^{th} output on the inputs $x_1, \dots, x_{j-1}$.

An example of MADE is illustrated in Figure 3.13. In practice, when the network is initialized, a color is assigned to each neuron uniformly at random in $\{1, \dots, n-1\}$ and the masks are generated accordingly (which remain fixed afterward).

Of particular interests are *masked autoregressive flows* [Papamakarios et al., 2017], which are MADEs that enable sampling from the distributions.

Definition 42: Masked autoregressive flow

A *masked autoregressive flow* (MAF) refers to the procedure which allows to *sample* from a MADE $f : \mathcal{X}^n \rightarrow \mathcal{Y}^n$ that models a density p of a continuous distribution over $\mathcal{X}^n$ through a parameterizable univariate density function p' over $\mathbb{R}$.

Let $\mathbf{x}_I \in \mathbb{R}^n$ be any arbitrarily initial input vector.^a For any $\mathbf{x} \in \mathbb{R}^n$ and $i \in \{1, \dots, n-1\}$, we write $z_i = f(\mathbf{x}_{<i})$ for the i^{th} output of the function $f(\mathbf{x}')$ where $\mathbf{x}'$ is the concatenation of the first $i-1$ elements of $\mathbf{x}$ with the last $n-i+1$ elements of $\mathbf{x}_I$, i.e., $\mathbf{x}' = \langle \mathbf{x}_1, \dots, \mathbf{x}_{i-1}, \mathbf{x}_{I,i}, \dots, \mathbf{x}_{I,n} \rangle$. In particular, we have $z_1 = f(\mathbf{x}_{<1})$.^b

Then, generating a sample $\mathbf{x} \sim p$ can be done through the following recursive procedure: let $1 \leq i \leq n$,

$$\mathbf{x}_i = x \quad \text{where} \quad x \sim p'(x \mid z_i) \quad \text{and} \quad z_i = f(\mathbf{x}_{<i}).$$

^ae.g., take $\mathbf{x}_I = \mathbf{0} = \langle 0, \dots, 0 \rangle \in \mathbb{R}^n$.

^bNotice that z_1 is completely independent of the input (cf. Figure 3.13). Also, notice that the output z_i is only dependent on inputs $\mathbf{x}_1, \dots, \mathbf{x}_{i-1}$, which ensures the autoregressive property. Then, the i^{th} output z_i of $f(\mathbf{x}_{<i})$ is the same *whatever the initial vector* $\mathbf{x}_I \in \mathbb{R}^n$ *chosen*.

A notable choice of densities p' are reparameterizable ones (such as those belonging to the loc-scale family), which allows for optimizing the network by backpropagating the gradients through the sampling procedure. For instance, Papamakarios et al. [2017] chose the normal distribution so that $f(\mathbf{x}_{<i})$ produces a mean and a variance for each $i \leq n$.

To model the transition function $\bar{\mathbf{P}}$ and further produce multivariate, relaxed Bernoulli distributions, we use a MAF so that f produces the logits α of the Bernoulli distribution. Concretely, recall that generating relaxed Bernoullis for a given temperature $\lambda > 0$ amounts to drawing a logistic sample located in α/λ and scaled to $1/\lambda$, where a sigmoid is applied in downstream (cf. Section 2.4 for details). Then, the MADE is trained to produce the logits of a Bernoulli distribution via the following recursion: let $i \in \{1, \dots, n\}$,

$$\mathbf{b}_i = \sigma(l_i) \quad \text{where} \quad l_i = \frac{\epsilon_i + \alpha_i}{\lambda}, \quad \epsilon_i \sim \text{Logistic}(0, 1), \quad \text{and} \quad \alpha_i = f(\mathbf{b}_{<i}). \quad (3.17)$$

Then, we use a MADE coupled with the MAF of Equation 3.17 to model the transition function $\bar{\mathbf{P}}$. To model the latent stationary distribution ξ_{π} , we employ a MADE to learn the distribution over the labels (the first $|\mathbf{AP}|$ bits of the latent states). The remaining bits are uniformly sampled at random, with their logits fixed to 0. In essence, the network

modeling $\bar{\xi}_\pi$ focuses on estimating the marginal dynamics of the labels, while the fixed remaining bits ensure a fairly distributed latent space.

Remark 24: MADE and conditional distributions

Recall that $\bar{\mathbf{P}}(\cdot \mid \bar{s}, \bar{a})$ is a distribution conditioned by $\bar{s} \in \bar{\mathcal{S}}$ and $\bar{a} \in \bar{\mathcal{A}}$. To model conditional distributions with MADEs, one simply needs to concatenate the conditioning to the input of the network and keep all the connections from the conditioning to the output layer.

3.5 The Latent Metric

In this section, we elaborate on the latent metric $\bar{d}$ used in the regularizers L_{ξ_π} and L_p^π . When considering relaxed distributions, the latent metric $\bar{d}$ should be adapted to the *continuous* relaxations, ensuring that $\bar{d}$ accounts for the discretization effect when the temperature approaches zero.

Precisely, in the following, we show that considering the Euclidean distance for $\vec{d}$ and $\bar{d}$ in the latent space, to optimize the regularizers L_{ξ_π} and L_p^π , is *Lipschitz equivalent* to considering a *continuous λ -relaxation of the discrete metric*. Consequently, this means it is consistently sufficient to enforce the 1-Lipschitzness via the gradient penalty approach of Gulrajani et al. [2017], whose guarantees hold when the Euclidean distance is the metric used in the latent space.

Lemma 19: Continuous relaxation of the discrete metric

Let d be the usual Euclidean distance and $d_\lambda : [0, 1]^n \times [0, 1]^n \rightarrow [0, 1)$, $\langle \mathbf{x}, \mathbf{y} \rangle \mapsto d(\mathbf{x}, \mathbf{y}) / (\lambda + d(\mathbf{x}, \mathbf{y}))$ for $\lambda \in (0, 1]$ and $n \geq 1$, then d_λ is a distance metric. Furthermore, d_λ converges to $\mathbb{1}_\neq$ as λ goes to zero.

Proof. The function d_λ is a metric if and only if it satisfies the following axioms (cf. Definition 31):

- (i) *Identity of indiscernibles:* We have that $d_\lambda(\mathbf{x}, \mathbf{y}) = \frac{d(\mathbf{x}, \mathbf{y})}{\lambda + d(\mathbf{x}, \mathbf{y})} = 0$ if and only if $d(\mathbf{x}, \mathbf{y}) = 0$. Since d is a metric, $d(\mathbf{x}, \mathbf{y}) = 0$ if and only if $\mathbf{x} = \mathbf{y}$. Then, by transitivity, $d_\lambda(\mathbf{x}, \mathbf{y}) = 0$ if and only if $\mathbf{x} = \mathbf{y}$.

- (ii) *Symmetry:* again, since d is a metric, we have

$$d_\lambda(\mathbf{x}, \mathbf{y}) = \frac{d(\mathbf{x}, \mathbf{y})}{\lambda + d(\mathbf{x}, \mathbf{y})} = \frac{d(\mathbf{y}, \mathbf{x})}{\lambda + d(\mathbf{y}, \mathbf{x})} = d_\lambda(\mathbf{y}, \mathbf{x}).$$

(iii) *Triangle inequality*: let $x, y, z \in [0, 1]^n$, the triangle inequality holds whenever

$$\begin{aligned}
 & d_\lambda(x, y) + d_\lambda(y, z) \geq d_\lambda(x, z) \quad (3.18) \\
 \Leftrightarrow & \frac{d(x, y)}{\lambda + d(x, y)} + \frac{d(y, z)}{\lambda + d(y, z)} \geq \frac{d(x, z)}{\lambda + d(x, z)} \\
 \Leftrightarrow & \frac{\lambda d(x, y) + \lambda d(y, z) + 2d(x, y)d(y, z)}{\lambda^2 + \lambda d(x, y) + \lambda d(y, z) + d(x, y)d(y, z)} \geq \frac{d(x, z)}{\lambda + d(x, z)} \\
 \Leftrightarrow & (\lambda d(x, y) + \lambda d(y, z) + 2d(x, y)d(y, z)) \cdot (\lambda + d(x, z)) \\
 & \geq d(x, z) \cdot (\lambda^2 + \lambda d(x, y) + \lambda d(y, z) + d(x, y)d(y, z)) \\
 \Leftrightarrow & \lambda^2 d(x, y) + \lambda^2 d(y, z) + 2\lambda d(x, y)d(y, z) + \\
 & \quad \lambda d(x, y)d(x, z) + \lambda d(y, z)d(x, z) + 2d(x, y)d(y, z)d(x, z) \\
 & \geq \lambda^2 d(x, z) + \lambda d(x, y)d(x, z) + \lambda d(y, z)d(x, z) + d(x, y)d(y, z)d(x, z) \\
 & \quad \text{(since } \lambda > 0 \text{ and } \text{Im}(d) \in [0, \infty)) \\
 \Leftrightarrow & \lambda^2 d(x, y) + \lambda^2 d(y, z) + 2\lambda d(x, y)d(y, z) + d(x, y)d(y, z)d(x, z) \\
 & \geq \lambda^2 d(x, z) \quad (3.19)
 \end{aligned}$$

Since d is a metric, it satisfies the triangle inequality:

$$\lambda^2 d(x, y) + \lambda^2 d(y, z) \geq \lambda^2 d(x, z). \quad (3.20)$$

Furthermore, $\text{Im}(d) \in [0, \infty)$, which means that

$$2\lambda d(x, y)d(y, z) + d(x, y)d(y, z)d(x, z) \geq 0 \quad (3.21)$$

By Equations 3.20 and 3.21, the inequality of Equation 3.19 holds. Furthermore, the fact that Equations 3.18 and 3.19 are equivalent yields the desired result.

Then, by (i), (ii), and (iii), d_λ is a distance metric. The fact that d_λ converges to $1_\neq$ as λ goes to zero is trivial by definition. $\square$

Lemma 20: Lipschitz equivalence of d and d_λ

Let d and d_λ be defined as in Lemma 19, then d and d_λ are Lipschitz-equivalent.

Proof. Distances d and d_λ are Lipschitz equivalent if and only if there exists some $a, b > 0$ such that for all $\mathbf{x}, \mathbf{y} \in [0, 1]^n$,

$$\begin{aligned} a \cdot d(\mathbf{x}, \mathbf{y}) &\leq d_\lambda(\mathbf{x}, \mathbf{y}) \leq b \cdot d(\mathbf{x}, \mathbf{y}) \\ \iff a \cdot d(\mathbf{x}, \mathbf{y}) &\leq \frac{d(\mathbf{x}, \mathbf{y})}{\lambda + d(\mathbf{x}, \mathbf{y})} \leq b \cdot d(\mathbf{x}, \mathbf{y}) \\ \iff a &\leq \frac{1}{\lambda + d(\mathbf{x}, \mathbf{y})} \leq b. \end{aligned}$$

Taking $a = \frac{1}{\lambda + \sqrt{n}}$ and $b = \frac{1}{\lambda}$ yields the result. $\square$

Corollary 5: Proper scale factor for the regularizers

For all $\beta \geq 1/\lambda$, $s \in \mathcal{S}$, $a \in \mathcal{A}$, $\bar{s} \in \bar{\mathcal{S}}$, and $\bar{a} \in \bar{\mathcal{A}}$, we have

- (i) $\mathcal{W}_{d_\lambda}(\mathcal{T}_\pi, \tilde{\xi}_\pi) \leq \beta \cdot \mathcal{W}_d(\mathcal{T}_\pi, \tilde{\xi}_\pi)$; and
- (ii) $\mathcal{W}_{d_\lambda}(\phi\mathbf{P}(\cdot | s, a), \bar{\mathbf{P}}(\cdot | \bar{s}, \bar{a})) \leq \beta \cdot \mathcal{W}_d(\phi\mathbf{P}(\cdot | s, a), \bar{\mathbf{P}}(\cdot | \bar{s}, \bar{a}))$.

Proof. Given any metric d^* , denote by Lipsch_d^K the set of K -Lipschitz functions w.r.t. d^* , i.e., $\text{Lipsch}_d^K = \{f : f(\mathbf{x}) - f(\mathbf{y}) \leq K \cdot d^*(\mathbf{x}, \mathbf{y})\}$. By the Lipschitz equivalence of d and d_λ , we know that $d_\lambda(\mathbf{x}, \mathbf{y}) \leq \beta \cdot d(\mathbf{x}, \mathbf{y})$ when $\beta \geq 1/\lambda$ (see the proof of Lemma 20). Then, it is clear that $\text{Lipsch}_{d_\lambda}^1 \subseteq \text{Lipsch}_d^\beta$. By the dual formulation of the Wasserstein distance (cf. Theorem 17), we have for any distributions P, Q that

$$\begin{aligned} \mathcal{W}_{d_\lambda}(P, Q) &= \sup_{f \in \text{Lipsch}_{d_\lambda}^1} \mathbb{E}_{x \sim P} f(x) - \mathbb{E}_{x \sim Q} f(x) \\ &\leq \sup_{f \in \text{Lipsch}_d^\beta} \mathbb{E}_{x \sim P} f(x) - \mathbb{E}_{x \sim Q} f(x) \\ &= \sup_{f \in \text{Lipsch}_d^1} \mathbb{E}_{x \sim P} [\beta \cdot f(x)] - \mathbb{E}_{x \sim Q} [\beta \cdot f(x)] = \beta \cdot \mathcal{W}_d(P, Q). \end{aligned}$$

We pass from the penultimate to the last line by constructing a β -Lipschitz function from a 1-Lipschitz function: let $f \in \text{Lipsch}_d^1$, we have $f(\mathbf{x}) - f(\mathbf{y}) \leq d(\mathbf{x}, \mathbf{y})$ if and only if $\beta \cdot f(\mathbf{x}) - \beta \cdot f(\mathbf{y}) \leq \beta \cdot d(\mathbf{x}, \mathbf{y})$. Then, $\beta \cdot f$ is a β -Lipschitz function.

Substituting P, Q to the desired functions yields the result. $\square$

Then, choosing the hyperparameter $\beta \geq 1/\lambda$ in the WAE-MDP objective function ensures that minimizing the β -scaled regularizers w.r.t. d also minimizes the regularizers w.r.t. the λ -relaxation d_λ , collapsing to the discrete distribution in the zero-temperature limit.

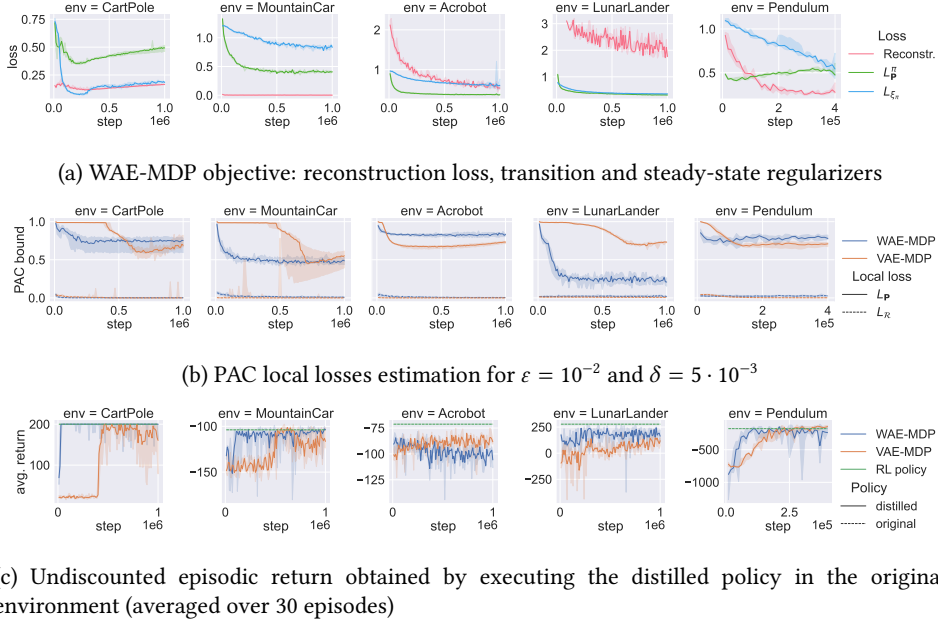


Figure 3.14: Evaluation of WAE-MDPs compared to VAE-MDPs. For each environment, we trained five different instances of the models with different random seeds: the solid line is the median and the shaded interval the interquartile range.

3.6 Experiments

We evaluate the quality of latent space models learned and policies distilled through WAE-MDPs. The results are depicted in Figure 3.14. We used the same original deep RL policies as those used for evaluating our VAE-MDPs: DQN policies on discrete, and SAC on continuous action spaces. This evaluation consists of: (a) the WAE-MDP training metrics, (b) the abstraction and representation quality via the PAC bounds on L_R and L_P presented in Section 1.3, (c) the distilled policy performance, and finally (d) the verification of time-to-failure properties in the latent models.

The confidence metrics and performance of the distilled policies are compared with those of VAE-MDPs. Additional details on our experiments can be found in Appendix B.

Learning metrics. To optimize the WAE-MDP objective, we use a simple experience replay buffer (with a large capacity; 10^6 samples) to emulate the stationary distribution, as well as minibatches of size 128 for approximating expectations through Monte Carlo

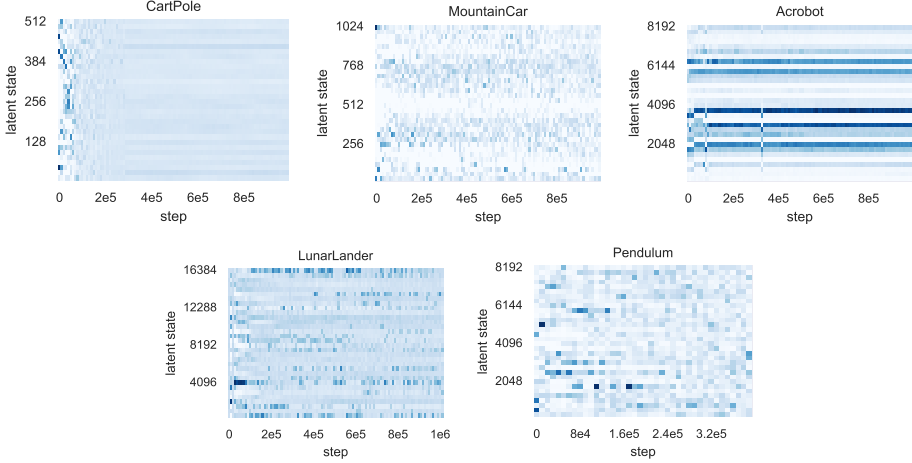


Figure 3.15: Latent space distribution along training steps. The intensity of the blue hue corresponds to the frequency of latent states produced by ϕ during training (compare with Figure 3.6).

estimations. The objective (Fig. 3.14a) is a weighted sum of the reconstruction loss and the two Wasserstein regularizers. We used different scale factor β for each regularizer. Posterior collapse is not observed, naturally avoided in WAEs [Tolstikhin et al., 2018], which reflects that the latent space is consistently distributed (see Section 2.6 for a discussion and a concrete illustration of collapsing issues occurring in VAE-MDPs). To illustrate that, we report the distribution of latent states produced by ϕ along the training steps (Figure 3.15). The plots reveal that the latent space is fairly distributed during training, for each environment.

Optimizing the objective (Figure 3.14a) effectively allows for minimizing the local losses (Figure 3.14b) and recovering the performance of the original policy (Figure 3.14c). We stress that, in contrast to VAE-MDPs, we do not use any annealing schemes, extra entropy regularizers, or sophisticated sampling procedures such as prioritized experience replay to distill the policies.

Local losses. For V- and WAEs, we formally evaluate PAC upper bounds on $L_{\mathcal{R}}$ and $L_{\mathcal{P}}$. The lower the local losses, the closer $\mathcal{M}$ and $\overline{\mathcal{M}}$ are in terms of behaviors induced by $\bar{\pi}$. Our bounds quickly converge compared to VAEs, where the convergence is slow (cf. CartPole, MountainCar, and LunarLander) and possibly unstable (cf. CartPole and MountainCar). Finally, *when the distilled policies offer comparable performance*

Environment	step (10^5)	S	$\mathcal{A}$	$ \bar{S} $	$ \bar{\mathcal{A}} $	L_R (PAC)	L_P (PAC)	$ \hat{V}^\pi(s_t) - \bar{V}^\pi(s_t) $	$\bar{V}^\pi(s_t, \varphi)$
CartPole	1.2	$\subseteq \mathbb{R}^4$	$\{1, 2\}$	512	2	0.00499653	0.399636	3.71213	0.0316655
MountainCar	2.32	$\subseteq \mathbb{R}^2$	$\{1, 2\}$	1024	2	0.0141763	0.382323	2.83714	0
Acrobot	4.3	$\subseteq \mathbb{R}^6$	$\{1, 2, 3\}$	8192	3	0.0347698	0.649478	2.22006	0.0021911
LunarLander	3.2	$\subseteq \mathbb{R}^8$	$[-1, 1]^2$	16384	3	0.0207205	0.131357	0.0372883	0.0702039
Pendulum	3.7	$\subseteq \mathbb{R}^3$	$[-2, 2]$	8192	3	0.0266745	0.539508	4.33006	0.0348492

Table 3.2: Formal Verification of distilled policies. Values are computed for $\gamma = 0.99$ (lower is better).

(Figure 3.14c), our bounds are either close to or better than those of VAEs (except for Acrobot, where VAE- outperforms the WAE-MDP).

Distillation. The bisimulation guarantees are only valid for $\bar{\pi}$, the policy under which formal properties can be verified. Then, we expect $\bar{\pi}$ to achieve performance close to the original policy π when deployed in the RL environment. We evaluate the performance of $\bar{\pi}$ via the undiscounted episode return. We observe that the return approaches the original performance more rapidly for W- than VAEs. Specifically, WAEs converge quickly for all environments, whereas VAEs require focusing on stabilizing the learning process before proceeding to the distillation stage. The success in recovering the original performance emphasizes the representation quality guarantees induced by WAEs: when local losses are minimized, all original states that are embedded to the same representation are bisimilarly close. Distilling the policy over the new representation, albeit discrete and hence coarser, still achieves effective performance since ϕ keeps only what is important to preserve behaviors, and thus values. Furthermore, the distillation can remove some non-robustness obtained during RL: $\bar{\pi}$ prescribes the same actions for bisimilarly close states, whereas this is not necessarily the case for π .

Formal verification. To formally verify $\bar{\mathcal{M}}$, we used value iteration to compute discounted properties. Precisely, we verified *time-to-failure* properties φ , often used to check the failure rate of a system [Pnueli, 1977] by measuring whether the agent fails before the end of the episode. Although simple, such properties highlight the applicability of our approach on (constrained) reachability events translated into VLC’s strongly guarded formulae, which are building blocks to verify discounted linear properties in MDPs. In particular, we checked whether the agent

- (i) reaches an unsafe position or angle (CartPole, LunarLander), i.e., via the LTL formula $\neg \text{reset} \cup \text{fail}$ under its discounted VLC interpretation $\varphi = \lambda x.((\neg \text{reset} \wedge \gamma \cdot \odot x) \vee \text{fail})$;

- (ii) does not reach its goal position (MountainCar, Acrobot), i.e., $\neg \text{goal} \mathbf{U} \text{reset}$ with discounted interpretation $\varphi = \lambda x. ((\neg \text{goal} \wedge \gamma \cdot \circ x) \vee \text{reset})$; and
- (iii) does not reach and stay in a safe region of the system across episodes (Pendulum), i.e., $\Diamond(\neg \text{safe} \wedge \circ \text{reset})$ with discounted interpretation $\varphi = \lambda x. ((\neg \text{safe} \wedge \gamma \cdot \circ \text{reset}) \vee \gamma \cdot \circ x)$.⁸

Results are in Table 3.2: for each environment, we select the distilled policy that gives the best trade-off between performance (episode return) and abstraction quality (local losses). As an extra confidence metric, we report the value difference $|\hat{V}^{\bar{\pi}}(s_I) - \bar{V}^{\bar{\pi}}(\bar{s}_I)|$ obtained by executing $\bar{\pi}$ in $\mathcal{M}$ and $\bar{\mathcal{M}}$, where $\hat{V}^{\bar{\pi}}$ refers to the values in the original MDP estimated through Monte Carlo rollouts, while $\bar{V}^{\bar{\pi}}$ are the values computed in the latent MDP.

4 Concluding Discussion

In this chapter, we introduced a novel framework for distilling verifiable policies from deep RL agents through discrete latent space models. We showed that optimizing local reward and transition losses to learn the latent, abstract representation of the environment yields bisimulation guarantees between the original (unknown, continuous or very large) environment and the (simpler, discrete, abstract) latent model. Those bisimulation guarantees provide abstraction quality bounds that can be leveraged, coupled with PAC algorithms, to verify properties on the distilled policy, and lift the obtained guarantees to the original environment. Furthermore, they also yield representation quality guarantees, ensuring that states embedded to the same abstract representation are bisimilarly close.

We developed two models, VAE-MDPs and WAE-MDPs, to learn the latent space and distill the policy. Both are learned by executing the original RL policy in the environment. VAE-MDPs are based on the minimization of the Kullback Leibler divergence between the traces generated in the original and latent models. The minimization of this divergence can be realized through a variational lower bound on the log-likelihood of the traces produced, further inducing proxies to local losses. Although VAE-MDPs are able to distill RL policies onto a learned latent space, they suffer from posterior collapse and require additional annealing schemes, losses, and sophisticated sampling procedures to stabilize the learning process.

On the other hand, WAE-MDPs are based on the minimization of the Wasserstein distance between those traces. In contrast to VAE-MDPs, we showed that the learning objective can be directly linked to the local losses, allowing retrieving the bisimulation guarantees. In consequence, WAE-MDPs do not suffer from collapsing issues, and the

⁸note that $\Diamond(\neg \text{safe} \wedge \circ \text{reset})$ expresses the event of ending an episode in an unsafe state, meaning that either the agent never reached the safe region of the system or it reached it but eventually left it.

learning process is more stable and efficient, without the need for annealing schemes, additional regularization, or sophisticated sampling procedures.

Beyond distillation, our results, supported by Theorem 28, suggest that our WAE-MDP can be used as a *general latent space learner* for deep RL (cf. Remark 22), further opening possibilities to combine RL and formal methods *online* when no formal model is a priori known. For instance, this may be considered to address safety in RL with guarantees or decompose the standard synthesis pipeline within intractable environments into smaller, more manageable sub-tasks, abstracted via our algorithm.

Beyond Distillation

Learning to Believe in Partially Observable Environments through Reliable Latent Models

In the previous chapter, we hinted that the representation guarantees induced by our WAE-MDP model can be leveraged in the context of deep reinforcement learning, by using our algorithm as a general latent space learner. Indeed, the representation induced by our latent models yields bisimulation guarantees, ensuring that the latent space is a faithful representation of the environment that can be used to learn a policy.

In this chapter, we explore this idea further by considering more general environments, where the observation space is not necessarily *Markovian*. Precisely, we tackle the problem of learning in environments that are modeled by Markov decision processes, but where the observation perceived by the agent at each time step does not necessarily correspond to the full and true state of the environment. Such environments are referred to as *partially observable Markov decision processes* (POMDPs). These problems are common in many real-world applications, where the agent does not have access to the full state of the environment or only through noisy sensors. Notable examples include robotics [Lauri et al., 2023] and recommendation systems [Wu et al., 2021]. To illustrate the challenges of learning in POMDPs, let us consider the following example.

Example 18: Partially Observable MDP

Consider the simple grid world environment shown in Figure 4.1. In this environment, the agent 🐙 is only able to observe the color of the cell it occupies.

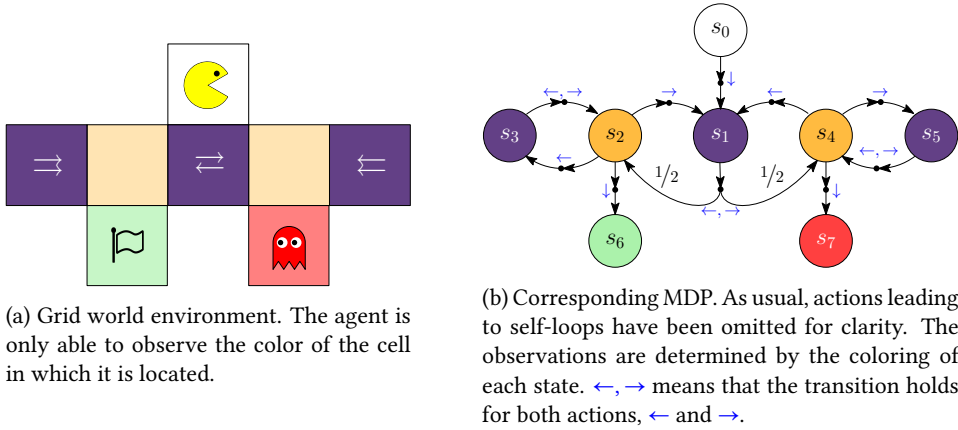


Figure 4.1: Simple grid world environment with partial observability. The observation function merely consists of the coloring of each state.

The grid contains shaded cells ■ where the outcome of the agent’s actions is uncertain: in s_1 , the agent has a probability of $1/2$ to move to the right and a probability of $1/2$ to move to the left, whatever the action chosen between $\leftarrow$ and $\rightarrow$. In the shaded cells at either end (s_3 and s_5), the agent will always move to the right or left, respectively, regardless of the action chosen ($\leftarrow$ or $\rightarrow$). The agent’s objective is to reach the flag, located in a cell with a green color ■. If the agent enters the red cell ■, it is caught by an adversary ■ and becomes trapped in a dead end. Consider a policy π that is solely conditioned by the current observation of the agent (i.e., the color of the cell). Clearly, stationary policies are not sufficient for seeking the optimal behavior in this environment: $\pi(\cdot \mid \text{orange})$ must prescribe $\downarrow$ in states colored in orange in order to reach the flag. However, if the agent is in s_4 , this choice leads to a dead end. Thus, π is clearly sub-optimal since it only allows reaching the target with probability $1/2$.

Then, the agent must base its decisions on the *history* of actions executed and observations perceived. For instance, once in a state colored in orange ■ (i.e., s_2 or s_4), the agent could play $\leftarrow$ a finite number of times $n > 0$ to increase its chance of being in s_2 , and then play $\downarrow$ after a sufficient number of steps. The related policy

can be defined as the deterministic mapping

$$\pi(a \mid \text{history} = \downarrow(\blacksquare \leftarrow \blacksquare \leftarrow)^i) = \begin{cases} \mathbb{1}\{a = \leftarrow\} & \text{if } i < n \\ \mathbb{1}\{a = \downarrow\} & \text{otherwise.} \end{cases}$$

Note that the size of the history space scales exponentially with the number of steps, so one could hardly rely on histories to define policies in general environments.

More compactly, the agent can maintain a *belief* over the state space, i.e., a probability distribution over the possible states occupied by the agent. For instance, after having executed $\leftarrow$ in the initial state and then $\leftarrow$ in the shaded cell $\blacksquare$, the agent has for belief $b_1 = \{s_2 \mapsto 1/2, s_4 \mapsto 1/2\}$.^a Then, by executing the action $\leftarrow$ twice, the agent increases its belief to $b_2 = \{s_2 \mapsto 3/4, s_4 \mapsto 1/4\}$, and so on. Once the belief of being in s_2 is sufficient, the agent can play $\downarrow$ to reach the flag. Given an error $\varepsilon > 0$, an ε -optimal policy for the reachability objective can thus be merely defined as

$$\pi(a \mid \text{belief} = b) = \begin{cases} \mathbb{1}\{a = \leftarrow\} & \text{if } b(s_4) > \varepsilon \\ \mathbb{1}\{a = \downarrow\} & \text{otherwise.} \end{cases}$$

^afor the sake of presentation, we omit the states that are not part of the support.

As illustrated in Example 18, in a POMDP, the agent perceives an imperfect observation of the state that does not suffice as a conditioning signal for an optimal policy. Instead, optimal policies must consider the complete interaction history. However, as the space of possible histories grows exponentially with the episode length, conditioning policies on histories is typically intractable. An alternative approach is the concept of *belief states*, which represent probability distributions over states derived from the agent’s history. Beliefs are a sufficient statistic of the history [Kaelbling et al., 1998] but the computation of their closed-form expression requires access to a model of the environment and is in general intractable.

To overcome those challenges, many state-of-the-art algorithms compress the history into a fixed-size vector with the help of *Recurrent Neural Networks* (RNNs) [Hausknecht and Stone, 2015] and *hope* to obtain this way a suitable representation of the history that could play the role of the belief. Nonetheless, this may lead to information loss, resulting in suboptimal policies. To improve the likelihood of obtaining a sufficient statistic, RNNs can be combined with regularization techniques, including generative models [Chen et al., 2022; Hafner et al., 2020, 2021], particle filtering [Igl et al., 2018; Ma et al., 2020], and predicting distant observations [Gregor et al., 2018, 2019]. However, *none of these techniques guarantee that the representation of histories induced by RNNs is suitable for optimizing the return*. Additionally, many algorithms assume that beliefs are

simple distributions (e.g., Gaussian), which limits their applicability [Gregor et al., 2018; Lee et al., 2020; Hafner et al., 2021].

In this chapter, we build upon Wasserstein Auto-encoded MDPs (WAE-MDPs, see Chapter 3, Section 3) to propose the *Wasserstein Belief Updater* (WBU), a model-based RL algorithm for POMDPs that allows learning the belief space over the unobservable states. Specifically, WBU learns an approximation of the belief update rule through a partially observable version of our WAE-MDP whose behaviors (expressed as expected return) are close to those of the original environment. Furthermore, we show that WBU is guaranteed to induce a suitable representation of the history to optimize the return. WBU consists of three components that are learned in a round-robin fashion: the model, the belief learner, and the policy (Figure 4.3).

Harnessing only histories to learn a model whose dynamics can be provably linked to the original unobservable environment poses a considerable challenge. Therefore, in the same spirit as the *Centralized Training with Decentralized Execution* paradigm in *multi-agent* RL (MARL) [Oliehoek et al., 2008], where leveraging additional information such as the true state of the environment is a common practice, *we assume that the POMDP states can be accessed during training*. While this might seem restrictive at first sight, this assumption is typically met in simulation-based training and can be applied in real-world settings such as robotics, where extra sensors can be used during training in a laboratory setting. This assumption is also compliant with typical software engineering approaches like model-driven design.

Our core contribution is the *development of a principled framework equipped with theoretical guarantees in the context of RL within partial observability*. While state-of-the-art algorithms primarily concentrate on enhancing the overall return – potentially resulting in substantial performance gains – we contend that performance is not the exclusive goal and that possessing guarantees is equally important, as the balance

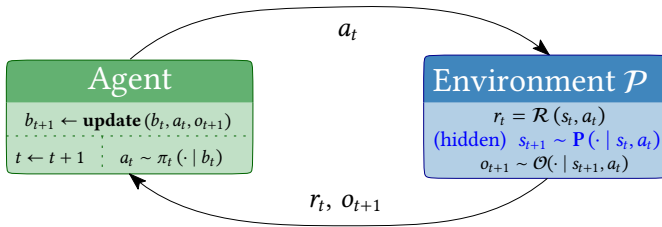


Figure 4.2: RL loop in POMDPs. The agent interacts with the environment by executing actions and receiving observations and rewards as feedback. At each time step, the agent maintains a belief over the state space which is updated based on the action taken and the observation received. This enables learning a policy that maps beliefs to actions.

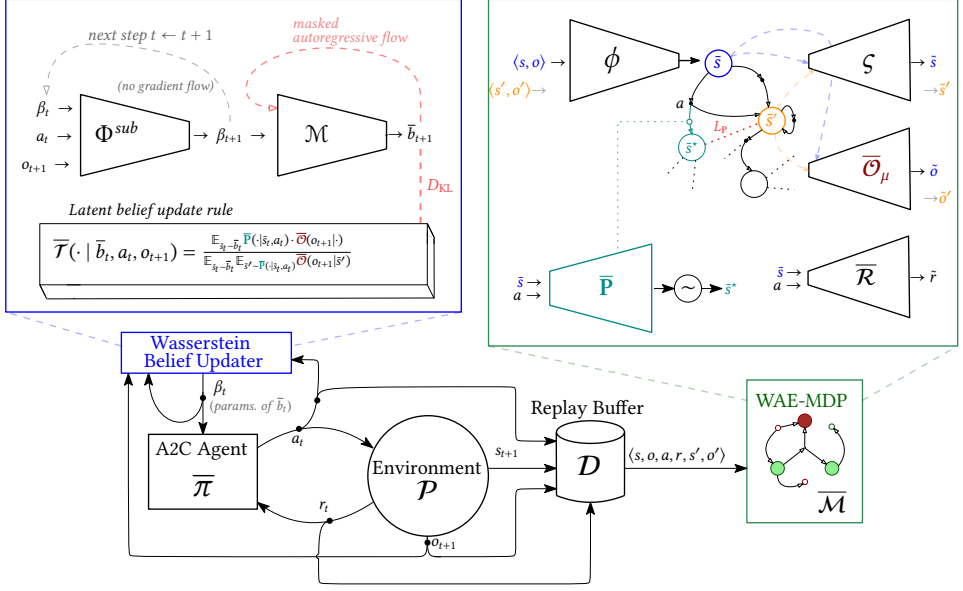


Figure 4.3: The *WBU framework*. Learning the different components is done in a round-robin fashion. The **WAE-MDP** learns from data collected by the RL agent and stored in a Replay Buffer. **WBU** uses the transition function $\bar{\mathbf{P}}$ and observation decoder $\bar{\mathcal{O}}$ of the **WAE-MDP** to learn to approximate the belief update rule. The agent learns a policy conditioned on the resulting *sub-belief* β_t (i.e., the parameters of the latent belief $\bar{b}_t$).

between these two aspects varies based on the specific application. By tackling POMDPs with a formal approach, we offer theoretical guarantees that other methods cannot provide: we ensure that *our latent model is able to replicate the dynamics of the original, partially observable environment*, which further yields a belief representation suitable for learning the value function.

While learning the latent model of the POMDP via a WAE-MDP — whose bisimulation bounds intuitively lead to our guarantees — we maintain a belief distribution over the latent state space via a *belief update network*: we minimize its distance to the exact belief update rule, through a tractable variational proxy. To allow for complex belief distributions, we use an autoregressive flow (Definition 42). In contrast to SOTA algorithms, the beliefs are only optimized towards accurately replicating its update rule. While we call recursively the belief network to maintain the belief distribution, we do not back-propagate through time and thus implement it as a simple (masked) feed-forward

network. The policy is learned on the latent belief space using a vector integrating the parameters of the belief distribution. Our experimental results are promising and show the ability of our algorithm to learn to encode the history into a representation useful to learn a policy, *without using RNNs*.

Other related work. DVRL [Igl et al., 2018] extends A2C [Mnih et al., 2016] (see Chapter 2, Section 2.3) combined with RNNs (R-A2C) with auxiliary losses aiming to learn beliefs via a variational auto-encoder and particle filtering, but it lacks guarantees. DVRL further assumes independent normal distributions for beliefs, limiting its applicability. In the same spirit as WBU, FORBES [Chen et al., 2022] uses normalizing flows to learn complex distributions but learn policies conditioned on latent states, which is sub-optimal as the state distribution is merely approximated with a single sample. Some works focus on specific POMDP types, like compact image representations (e.g., visual motor tasks, Lee et al., 2020) or states masked with Gaussian noise [Wang and Tan, 2021].

While accessing states is common in MARL, it is less common in single-agent. This has been explored in kernel-POMDPs [Nishiyama et al., 2012], and more recently by Lee et al. [2023] to provide results oriented towards sample complexity in the tabular case, as well as regret bounds when particular types of function approximators are used. Leveraging additional information available during training (not necessarily the states) has also been explored by Lambrechts et al. [2023], but RNNs remain crucial while no abstraction nor representation quality guarantee is provided. Several complexity results and regret bounds were developed for sub-classes of POMDPS in finite-horizon, such as when the observability is well-behaved (weakly revealed POMDPs, Liu et al., 2022b), or when the sequential decision making problem admits low-rank structure [Huang et al., 2024]. On the contrary, our results hold for general POMDPs; precisely, our constraints rather rely on the training procedure and not on the nature of the POMDP.

Finally, we note that RQL-AIS [SeyedSalehi et al., 2023] presents an analysis of recurrent Q -learning in the tabular setting, which also includes value difference bounds where the latent state consists of the (discrete) hidden state of an RNN. While theoretically interesting, their bounds are generally intractable as they rely on global metrics, requiring them to be minimized over the whole history and action space. This limitation is made clear in their algorithm as they use the local version of those metrics as regularization losses. In our case, because we focus on on-policy metrics (local losses) our theoretical guarantees hold in practice.

Origin of the results. The content of this chapter is novel in its presentation and is based on the article [Avalos et al., 2024], including additional details and discussions. The representation quality result (Theorem 31) is extended in this chapter to *any latent policy* (whereas the original result was restricted to the optimal policies of the latent model). The error term is not necessary in this version and the proof has been simplified.

1 Decision Making with Imperfect Observations

In this section, we formally introduce the notion of partially observable Markov decision processes (POMDPs), histories and the concept of belief states.

Remark 25: Dirac measures

In the following, we may consider the *Dirac delta function* δ as a *measure*. For any complete, separable space $\mathcal{X}$ and point $x \in \mathcal{X}$, we denote the *Dirac measure* with impulse x by $\delta_x \in \Delta(\mathcal{X})$. We see δ_x as a measure that concentrates all its mass to the single point x : $\delta_x(E) = \int_E \delta_x(y) dy = \mathbb{1}\{x \in E\}$ for any $E \in \Sigma(\mathcal{X})$. Interestingly, the Dirac measure δ_x can be seen as the limit of a sequence of normal distributions centered in x with decreasing variance: $\delta_x = \lim_{\sigma \rightarrow 0} \mathcal{N}(\cdot; x, \sigma^2)$, where $\mathcal{N}(\cdot; x, \sigma^2)$ is the normal distribution with mean x and variance σ^2 .

Definition 43: Partially Observable MDPs

A POMDP is a tuple $\mathcal{P} = \langle \mathcal{M}, \Omega, \mathcal{O} \rangle$ where

- $\mathcal{M}$ is an MDP with state space S and action space $\mathcal{A}$;
- Ω is a set of *observations*; and
- $\mathcal{O} : S \times \mathcal{A} \rightarrow \Delta(\Omega)$ is an *observation function* that defines the distribution of observations that may occur when the MDP $\mathcal{M}$ transitions to a state upon the execution of a particular action. We assume that $\mathcal{O}$ admits a density function, denoted by $p_{\mathcal{O}}$.

An agent in $\mathcal{P}$ actually interacts in $\mathcal{M}$, but *without directly observing the states of $\mathcal{M}$* : instead, the agent perceives observations, which yields *histories*, i.e., sequences of actions and observations $\langle a_{0:T-1}, o_{1:T} \rangle$ that can be associated to an (unobservable) trajectory $\langle s_{0:T}, a_{0:T-1} \rangle$ in $\mathcal{M}$, where $o_{t+1} \sim \mathcal{O}(\cdot \mid s_{t+1}, a_t)$ for all $t < T$.

The interaction of the agent within a POMDP is illustrated in Figure 4.2. In this chapter, we assume that the action space is finite.

Beliefs. Unlike in MDPs, where it is sufficient to make decisions based on the current state (cf. Chapter 2, Section 1), policies solely based on the current observation of $\mathcal{P}$ are not sufficient to optimize the return. Intuitively, due to the partial observability, the agent must take into account full histories in order to make an informed decision on its next action (cf. Example 18). Alternatively, the agent can maintain a *belief* $b_t \in \Delta(S)$ over the current state of $\mathcal{M}$ [Åström, 1965]. We denote by $\mathcal{B}$ the set of all reachable beliefs over S in $\mathcal{P}$. Given the next observation o_{t+1} , the next belief b_{t+1} is computed according to the

belief update function $\mathcal{T} : \mathcal{B} \times \mathcal{A} \times \Omega \rightarrow \mathcal{B}$, where $\mathcal{T}(b_t, a_t, o_{t+1}) = b_{t+1}$ if and only if given any measurable subset $\mathcal{X}' \in \Sigma(S)$, the next belief is given by

$$b_{t+1}(\mathcal{X}') = \frac{\int_{\mathcal{X}'} \int_S p_{\mathcal{O}}(o_{t+1} \mid s', a_t) d\mathbf{P}(s' \mid s, a_t) db_t(s)}{\int_{S \times S} p_{\mathcal{O}}(o_{t+1} \mid s', a_t) d\mathbf{P}(s' \mid s, a_t) db_t(s)}. \quad (4.1)$$

Each belief b_{t+1} constructed this way is a *sufficient statistic* for the history $\langle a_{0:t}, o_{1:t+1} \rangle$ to optimize the return [Kaelbling et al., 1998]. We write $\mathcal{T}^*(a_{0:t}, o_{1:t+1})$ for the recursive application of $\mathcal{T}$ along the history, i.e.,

$$\mathcal{T}^*(a_{0:t}, o_{1:t+1}) = \mathcal{T}(\cdot, a_t, o_{t+1}) \circ \mathcal{T}^*(a_{0:t-1}, o_{1:t})$$

for for $t > 0$ and $\mathcal{T}^*(a_0, o_1) = \mathcal{T}(\delta_{s_I}, a_0, o_1)$. The belief update rule derived from $\mathcal{T}$ allows to formulate $\mathcal{P}$ as an infinitely large¹ *belief MDP* $\mathcal{M}_{\mathcal{B}} = \langle \mathcal{B}, \mathcal{A}, \mathbf{P}_{\mathcal{B}}, \mathcal{R}_{\mathcal{B}}, b_I, \gamma \rangle$, where the density of $\mathbf{P}_{\mathcal{B}}$ is given by

$$\mathbf{p}_{\mathcal{B}}(b' \mid b, a) = \mathbb{E}_{s \sim b} \mathbb{E}_{s' \sim \mathbf{P}(\cdot \mid s, a)} \mathbb{E}_{o' \sim \mathcal{O}(\cdot \mid s', a)} \delta_{\mathcal{T}(b, a, o')}(b'); \quad (4.2)$$

$\mathcal{R}_{\mathcal{B}}(b, a) = \mathbb{E}_{s \sim b} \mathcal{R}(s, a)$; and $b_I = \delta_{s_I}$. As for all MDPs, $\mathcal{M}_{\mathcal{B}}$ and any stationary policy for $\mathcal{M}_{\mathcal{B}}$ (thus conditioned on beliefs) induces a well-defined probability space over trajectories of $\mathcal{M}_{\mathcal{B}}$, which allows optimizing the expected return in $\mathcal{P}$ (cf. Theorem 5).

Value Functions. In consequence, we obtain the value function of the POMDP $\mathcal{P}$ by considering the values obtained in the belief MDP $\mathcal{M}_{\mathcal{B}}$. Precisely, for any stationary policy $\pi : \mathcal{B} \rightarrow \Delta(\mathcal{A})$ for $\mathcal{M}_{\mathcal{B}}$ and belief $b \in \mathcal{B}$, the value of the belief b for the particular policy π is given by

$$\begin{aligned} V^{\pi}(b) &= \mathcal{R}_{\mathcal{B}}(b, a) + \gamma \cdot \mathbb{E}_{b' \sim \mathbf{P}_{\mathcal{B}}(\cdot \mid b, a)} V^{\pi}(b') \\ &= \mathbb{E}_{s \sim b} \left[\mathcal{R}(s, a) + \gamma \cdot \mathbb{E}_{s' \sim \mathbf{P}(\cdot \mid s, a)} \mathbb{E}_{o' \sim \mathcal{O}(\cdot \mid s', a)} V^{\pi}(\mathcal{T}(b, a, o')) \right]. \end{aligned}$$

Given any history $h \in (\mathcal{A} \cdot \Omega)^*$, we write $V^{\pi}(h)$ as a shorthand for $V^{\pi}(\mathcal{T}^*(h))$. Thus, the value of the history h under the policy π corresponds to the return expected by starting to execute π after having experienced this history h .

¹even if S is finite, there is an infinite number of measures in $\mathcal{B}$. As remarked by Bork et al. [2022], when the observation space is finite, the reachable fragment of the belief space is countable since each belief has at most $|\mathcal{A} \times \Omega|$ successors.

2 Learning the Dynamics

In the remainder of this chapter, we assume that the agent operates within a POMDP. In an RL setting, the former has no explicit access to the environment dynamics: instead, it reinforces its behaviors through interactions and experiences without directly accessing the transition, reward, and observation functions of the environment (Figure 4.2).

The key of our approach lies in *granting the RL agent access to the true state of the environment during its training*, while *its perception of the environment is only limited to actions and observations when the learned policy is finally deployed*.

Assumption 9: Access to the state during training

In addition to the observation, the agent is able to observe the true state of the environment, but only during the training phase.

Remark 26: About the access of the states during training

Seemingly restrictive, Assumption 9 can actually be met in a broad range of training scenarios, in particular those relying on simulators, where one could merely consider the RAM as the state. Furthermore, there are multiple settings where the state is available during training but not during the execution. First, when the state is too large (e.g., RAM, images) to be processed by the agent in real-time (e.g., an agent running on low-power hardware). Then, in situations where the agent is trained within a *Simulation to Reality* framework, an example being the introduction of noise into the state space to create resilient policies intended for real-world deployment [Staessens, Tom, 2023]. Other settings compliant with this assumption include model-based design or model-predictive control, where a computational model is already available in the training phase. Finally, accessing the state could be costly [Bulychev et al., 2012], and while the cost might be acceptable in simulation or at training time, it might not be the case in deployment time (e.g., embedded systems, deploying multiple agents running in independent environments, or when a final product is shipped).

For the rest of this chapter, fix the POMDP $\mathcal{P} = \langle \mathcal{M}, \Omega, \mathcal{O} \rangle$ with underlying MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathbf{P}, \mathcal{R}, s_I, \gamma \rangle$. Concretely, when the RL agent interacts in $\mathcal{P}$, we leverage this access to allow the agent to learn the dynamics of the environment, i.e., those of $\mathcal{M}$, as well as those related to the observation function $\mathcal{O}$. To do so, we learn an internal, explicit representation of the experiences gathered, through a latent space model. We then use this model as a “teacher” for the agent to make it learn how to perform its belief updates. Hence, acquiring an accurate environment model is crucial to learning a reliable

belief update function. In Section 2.4, we further demonstrate that the resulting model is guaranteed to closely replicate the original environment behaviors. The trick we use to learn such a model is to reason on an equivalent POMDP, where the underlying MDP is refined to encode all the crucial dynamics.

2.1 The Latent POMDP Dynamics

We enable learning the dynamics of $\mathcal{P}$ via a WAE-MDP by considering the *refined* POMDP $\mathcal{P}^\dagger = \langle \mathcal{M}_\Omega, \Omega, \mathcal{O}^\dagger \rangle$, which is defined as follows.

- (i) The underlying MDP $\mathcal{M}_\Omega$ is refined to encode the observations, i.e.,

$$\mathcal{M}_\Omega = \langle S_\Omega, \mathcal{A}, \mathbf{P}_\Omega, \mathcal{R}_\Omega, \langle s_I, o_I \rangle, \gamma \rangle,$$

where

- the state is refined with the observation space, i.e., $S_\Omega = S \times \Omega$;
- the refined transition function encodes both the MDP and the observation dynamics: for any $S \in \Sigma(S)$ and $O' \in \Sigma(\Omega)$,

$$\mathbf{P}_\Omega(S' \times O' \mid s, o, a) = \int_{S'} \int_{O'} p_\Theta(o' \mid s', a) \, \mathrm{d}o' \, \mathrm{d}\mathbf{P}(s' \mid s, a),$$

- the reward function is independent of the observation: $\mathcal{R}_\Omega(\langle s, o \rangle, a) = \mathcal{R}(s, a)$; and
- o_I is an arbitrary observation from Ω linked to the initial state s_I .

- (ii) The observation function $\mathcal{O}^\dagger : S_\Omega \rightarrow \Omega$ is the *deterministic* projection of the refined state on the observation space, with $\mathcal{O}^\dagger(\langle s, o \rangle) = o$ for all $s \in S, o \in \Omega$.

The POMDPs $\mathcal{P}$ and $\mathcal{P}^\dagger$ are equivalent [Chatterjee et al., 2016]. Intuitively, the observation dynamics are captured by the transition function of the refined MDP, whose states are not observed by the agent. The latter still only perceive the same observations as in $\mathcal{P}$. Then, for any history, the value functions of the two POMDPs yield the same return. Nonetheless, the refined model $\mathcal{P}^\dagger$ notably uses a *deterministic* observation function $\mathcal{O}^\dagger$, which makes it easier to learn via an auto-encoder procedure.

Henceforth, we reduce the problem of learning a latent space model of $\mathcal{P}$ to learning a WAE-MDP from $\mathcal{M}_\Omega$. Precisely, we learn a latent MDP $\bar{\mathcal{M}} = \langle \bar{S}, \mathcal{A}, \bar{\mathbf{P}}, \bar{\mathcal{R}}, \bar{s}_I, \gamma \rangle$ linked to $\mathcal{M}_\Omega$ via the embedding $\phi : S_\Omega \rightarrow \bar{S}$. The latent MDP $\bar{\mathcal{M}}$ encodes the observation dynamics through $\bar{\mathbf{P}}$ and enables to learning the deterministic observation function $\mathcal{O}^\dagger$ through the decoder $\Psi : \bar{S} \rightarrow S_\Omega$, by decomposing the latter in two networks $\zeta : \bar{S} \rightarrow S$ and $\mathcal{O}_\mu : \bar{S} \rightarrow \Omega$, further yielding $\Psi(\bar{s}) = \langle \zeta(\bar{s}), \mathcal{O}_\mu(\bar{s}) \rangle$ for all $\bar{s} \in \bar{S}$. This way, the WAE-MDP learns all the components of $\mathcal{P}^\dagger$, the latter being equivalent to $\mathcal{P}$. With this model,

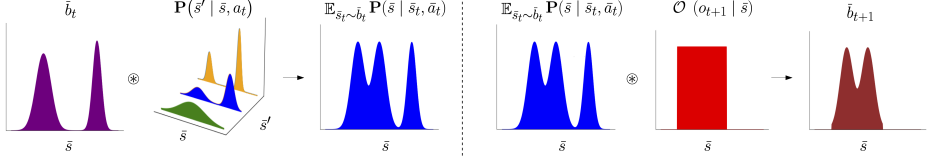


Figure 4.4: The belief update rule: (left) marginalization of the transition probability function $\bar{\mathbf{P}}$ according to the current belief $\bar{b}_t$; (right) filtering out the next states that do not produce the next observation o_{t+1} .

we construct a *latent POMDP* $\bar{\mathcal{P}} = \langle \bar{\mathcal{M}}, \Omega, \bar{\mathcal{O}} \rangle$, where $\bar{\mathcal{O}}$ is a Dirac measure with impulse $\bar{\mathcal{O}}_\mu$. Notably, WAE-MDPs yield finite latent spaces (Chapter 3, Section 3.4), which eases its theoretical analysis. We write $\bar{V}^\pi$ for the value function of any latent policy π for $\bar{\mathcal{P}}$.

Belief update rule. As with any POMDP, the belief update function $\bar{\mathcal{T}}$ of $\bar{\mathcal{P}}$ allows reasoning on the belief space $\bar{\mathcal{B}} \subseteq \Delta(\bar{\mathcal{S}})$ to optimize the return. Formally, at step $t \geq 0$, assume that the current latent belief is $\bar{b}_t \in \bar{\mathcal{B}}$. Then, when $a_t \in \mathcal{A}$ is executed and $o_{t+1} \in \Omega$ is observed, $\bar{b}_t$ is updated according to $\bar{b}_{t+1} = \bar{\mathcal{T}}(\bar{b}_t, a_t, o_{t+1})$ as follows, so that for any next state $\bar{s}_{t+1} \in \bar{\mathcal{S}}$,

$$\bar{b}_{t+1}(\bar{s}_{t+1}) = \frac{\mathbb{E}_{\bar{s}_t \sim \bar{b}_t} \bar{\mathbf{P}}(\bar{s}_{t+1} | \bar{s}_t, a_t) \cdot \bar{\mathcal{O}}(o_{t+1} | \bar{s}_{t+1})}{\mathbb{E}_{\bar{s}_t \sim \bar{b}_t} \mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot | \bar{s}_t, a_t)} \bar{\mathcal{O}}(o_{t+1} | \bar{s}')} . \quad (4.3)$$

As this latent belief update rule is at the heart of this chapter, it is useful to take a closer look at how it is constructed. Notice that Equation 4.3 can be merely obtained via a pure application of the Bayes rule:

$$\begin{aligned} \Pr(\bar{s}_{t+1} | \bar{b}_t, a_t, o_{t+1}) &= \frac{\Pr(o_{t+1} | \bar{b}_t, a_t, \bar{s}_{t+1}) \cdot \Pr(\bar{s}_{t+1} | \bar{b}_t, a_t)}{\Pr(o_{t+1} | \bar{b}_t, a_t)} \\ &= \frac{\Pr(o_{t+1} | \bar{b}_t, a_t, \bar{s}_{t+1}) \cdot \mathbb{E}_{\bar{s}_t \sim \bar{b}_t} \Pr(\bar{s}_{t+1} | \bar{s}_t, a_t)}{\Pr(o_{t+1} | \bar{b}_t, a_t)}, \end{aligned}$$

where $\Pr(\bar{s}_{t+1} | \bar{b}_t, a_t, o_{t+1}) = \bar{\mathcal{T}}(\bar{s}_{t+1} | \bar{b}_t, a_t, o_{t+1})$, $\Pr(o_{t+1} | \bar{b}_t, a_t, \bar{s}_{t+1}) = \bar{\mathcal{O}}(o_{t+1} | \bar{s}_{t+1})$, $\Pr(\bar{s}_{t+1} | \bar{s}_t, a_t) = \bar{\mathbf{P}}(\bar{s}_{t+1} | \bar{s}_t, a_t)$, and $\Pr(o_{t+1} | \bar{b}_t, a_t) = \mathbb{E}_{\bar{s}_t \sim \bar{b}_t} \mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot | \bar{s}_t, a_t)} \bar{\mathcal{O}}(o_{t+1} | \bar{s}')$.

We further illustrate the construction of the latent belief update rule in Figure 4.4. The update can be subdivided into three steps. First, the current belief distribution $\bar{b}_t$ is used to

marginalize the latent transition function $\bar{\mathbf{P}}$ over the believed latent states, to further infer the distribution over the possible next states. This first part corresponds to computing the probability of going to the next state $\bar{s}_{t+1}$ from the possible states $\bar{s}_t$ witnessed by the current belief $\bar{b}_t$. Second, the next observation o_{t+1} is used to filter the resulting distribution based on the observation function: the next latent states that do not share the observation o_{t+1} perceived at step $t + 1$ are filtered out according to $\bar{\mathcal{O}}(o_{t+1} \mid \bar{s}_{t+1})$. The third operation (not present in Figure 4.4) normalizes the output of the observation filtering to obtain a well-defined probability distribution.

Interestingly, notice that since the latent model has discrete spaces, each latent belief $\bar{b} \in \bar{\mathcal{B}}$ can be represented as a vector of probabilities from the simplex

$$\Delta^n = \left\{ \mathbf{x} \in \mathbb{R}^n : \sum_{i=1}^n x_i = 1 \text{ and } x_i \geq 0 \text{ for all } i = 1, \dots, n \right\}$$

with $n = |\bar{\mathcal{S}}|$. Then, we have for every $\bar{b}_1, \bar{b}_2 \in \Delta^n$ that $\mathcal{W}_{\bar{d}}(\bar{b}_1, \bar{b}_2) = d_{TV}(\bar{b}_1, \bar{b}_2) = \frac{1}{2} \cdot \|\bar{b}_1 - \bar{b}_2\|_1$, where $\|\cdot\|_1$ denotes the usual L_1 norm.

Latent policies. Given *any* history $h \in (\mathcal{A} \cdot \Omega)^*$, running a latent stationary policy $\bar{\pi} : \bar{\mathcal{B}} \rightarrow \Delta(\mathcal{A})$ in $\mathcal{P}$ is made possible by converting h into a latent belief $\bar{\mathcal{T}}^*(h) = \bar{b}$ and executing the action prescribed by $\bar{\pi}(\cdot \mid \bar{b})$. Training $\bar{\mathcal{M}}$ grants access to the dynamics required to update the belief through its closed form (Equation 4.3). However, marginalizing over the full latent space remains computationally intractable.²

As a solution, we propose to leverage the access to the dynamics of $\bar{\mathcal{M}}$ to learn a *latent belief encoder* $\Phi : \bar{\mathcal{B}} \times \mathcal{A} \times \Omega \rightarrow \bar{\mathcal{B}}$ that approximates the belief update function by minimizing

$$D(\bar{\mathcal{T}}^*(h), \Phi^*(h)) \tag{4.4}$$

for *some* discrepancy D and histories h drawn from *some* distribution. The belief encoder Φ thus enables to learn a policy $\bar{\pi}$ conditioned on latent beliefs to optimize the return in $\mathcal{P}$: *given the current history h , the next action follows $a \sim \bar{\pi}(\cdot \mid \Phi^*(h))$.*^a

^aWe define Φ^* as the recursive application of Φ along the history, in a similar way as $\bar{\mathcal{T}}^*$.

To offer theoretical guarantees,

- (i) the WAE-MDP reduction should induce a latent POMDP whose behaviors are close to those of $\mathcal{P}$, and
- (ii) the history representation induced by Φ should be suitable for optimizing the expected return in $\mathcal{P}$.

²In our experiments, we used up to 2^{20} latent states.

Clearly, the choice of the discrepancy and the history distribution are crucial. The rest of this section is dedicated to the theoretical analysis of the proposed framework, where we detail the required conditions (including discrepancy and history distribution) to ensure that the latent model is suitable for learning the belief update rule.

2.2 Stationarity over Histories

To yield the guarantees, we specifically target the *episodic* setting for drawing histories (cf. Chapter 2, Section 2.1 for a full discussion on this setting). In partially observable environments, this means that the underlying MDP is episodic and that *the agent is able to observe the reset state*.

Assumption 10: Episodic RL process in partially observable environments

We assume that the environment $\mathcal{P}$ is episodic, i.e., (i) the underlying MDP $\mathcal{M}$ is episodic (Definition 16), and (ii) there is a special observation $o_{reset} \in \Omega$ that can only be perceived in s_{reset} , so that $\mathcal{O}(o_{reset} \mid s_{reset}, a) = 1$ for all $a \in \mathcal{A}$.

Intuitively, as in the fully observable case, Assumption 10 implies the ergodicity of the system, which allows to derive a stationary distribution. Nonetheless, this time the distribution is over histories.

Proposition 1: Stationary distribution over histories

There exists a well-defined, unique stationary distribution $\mathcal{H}_{\bar{\pi}} \in \Delta((\mathcal{A} \cdot \Omega)^*)$ over histories likely to be produced when the latent policy $\bar{\pi}$ is lifted in $\mathcal{P}$.

Proof sketch. Build a *history unfolding* as the MDP whose state space consists of all histories and keeps track of the current history of $\mathcal{P}$ at any time of the interaction, until the reset state is observed, at which point the history is reset as well. The resulting MDP remains episodic and is equivalent to $\mathcal{P}$: the former mimics the behaviors of the latter under any latent policy $\bar{\pi}$. All episodic processes are *ergodic* (Theorem 7), which guarantees the existence of such a distribution in the history unfolding, further defining the stationary distribution over the histories of $\mathcal{P}$. $\square$

In the remainder of this subsection, we formally elaborate on and provide technical proofs for each assertion outlined in this sketch. Essentially, our goal is to construct a history unfolding which is *equivalent* to the original POMDP, when both operate under any given latent policy $\bar{\pi}$ lifted to the models by processing histories through the belief encoder Φ . This equivalence is formalized as a *bisimulation relation* (Definition 29) between the Markov chains respectively induced by the history unfolding and the belief

MDP of $\mathcal{P}$ when such a latent policy $\bar{\pi}$ is executed. Then, we show the existence of a unique stationary distribution over histories in the unfolding.

History unfolding. We define the *history unfolding* $\mathcal{M}_H$ as the MDP consisting of the tuple $\langle S_H, \mathcal{A}, \mathbf{P}_H, \star, \gamma \rangle$, where

- (i) the state space consists of the set of all the possible histories (i.e., sequence of actions and observations) that can be encountered in $\mathcal{P}$, i.e., $S_H = (\mathcal{A} \cdot \Omega)^*$, where we notably denote by ϵ the empty string, with $\mathcal{T}^*(\epsilon) = \delta_{s_t}$ (no action nor observation has been perceived yet);
- (ii) the transition function maps the current history to the belief space to infer the distribution over the next possible observations, i.e., the density of $\mathbf{P}_H$ is defined as follows: for all $h, h' \in S_H$ and $a \in \mathcal{A}$,

$$\begin{aligned} \mathbf{p}_H(h' | h, a) &= \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, a)} \mathbb{E}_{o' \sim \mathcal{O}(\cdot | s', a)} \delta_{h \cdot a \cdot o'}(h') & \text{if } \mathcal{T}^*(h) \neq \delta_{s_{\text{reset}}}, \text{ and}^3 \\ \mathbf{p}_H(\epsilon | h, a) &= 1 & \text{otherwise,} \end{aligned}$$

where we denote the concatenation of a, o' with the history $h = \langle a_{0:T-1}, o_{1:T} \rangle$ by $h \cdot a \cdot o'$, resulting in the history $\langle a_{0:T}, o_{1:T+1} \rangle$ so that $a_T = a$ and $o_{T+1} = o'$; and

- (iii) the reward function maps the history to the belief space as well, which enables to infer the expected rewards obtained in the states witnessed under this belief, i.e., $\mathcal{R}_H(h, a) = \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathcal{R}(s, a)$.

We want to show that $\mathcal{M}_H$ is equivalent to $\mathcal{M}_B$ (the Markovian formulation of $\mathcal{P}$) when a latent policy $\bar{\pi}$ is executed in both models by processing histories through the belief encoder Φ . This lifting is straightforward in $\mathcal{M}_H$: for any *history state* $h \in S_H$, h is processed by Φ to obtain the latent belief $\bar{b} = \Phi^*(h)$, which is then used to sample the next action according to $\bar{\pi}(\cdot | \bar{b})$. We denote the resulting policy for $\mathcal{M}_H$ by $\bar{\pi}^\star$, where $\bar{\pi}^\star(\cdot | h) = \bar{\pi}(\cdot | \Phi^*(h))$ for all $h \in S_H$.

Within $\mathcal{M}_B$, the lifting is more intricate. Indeed, one needs to infer the current latent belief $\bar{b}$ solely from the current belief state $b \in \mathcal{B}$ of $\mathcal{M}_B$, which is not straightforward. To do so, we use a Mealy machine (Property 1) to encode the lifted policy and keep track of the current *latent* belief through the memory of the lifted policy.

Before going further, we introduce a technical Lemma that will be useful to prove the equivalence between the two models. Intuitively, every state of the Mealy machine is updated according to the action executed from the current state of the environment. However, we need to synchronize the Mealy machine with the next state of the

³Notice that $\mathcal{T}^*(h) = \delta_{s_{\text{reset}}}$ whenever the history h ends with the observation o_{reset} , which is the sole possible observation perceived when entering the reset state s_{reset} .

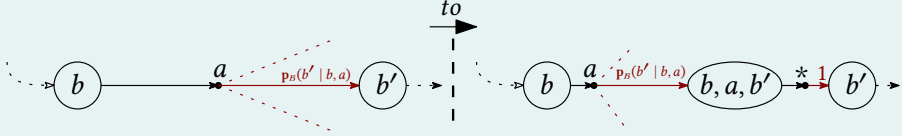
environment to ensure that the latent belief is correctly inferred. To do so, it suffices to (i) add a dummy state $\langle b, a, b' \rangle$ which is entered each time the system transitions from a belief b to another b' when an action a is chosen, before actually transitioning to b' , and (ii) changing the discount to $\sqrt{\gamma}$ to compensate for the addition of this extra transition. The following Lemma states that adding such a dummy transition to the belief MDP which allows for synchronizing the Mealy machine with the next belief state is harmless.

Lemma 21: Dummy states

Let $\mathcal{M}_B$ be the belief MDP of $\mathcal{P}$. Construct $\mathcal{M}'_B = \langle \mathcal{B}', \mathcal{A}, \mathbf{P}'_B, \mathcal{R}'_B, s_I, \sqrt{\gamma} \rangle$ so that $\mathcal{B}' = \mathcal{B} \cup (\mathcal{B} \times \mathcal{A} \times \mathcal{B})$ by modifying the transition density function of $\mathcal{M}_B$ to

$$\mathbf{p}'_B(\langle b, a, b' \rangle | b, a) = \mathbf{p}_B(b' | b, a), \quad \text{and} \quad \mathbf{p}'_B(\cdot | \langle b, a, b' \rangle, a') = \delta_{b'},$$

as well as the reward function to $\mathcal{R}'_B(b, a) = \mathcal{R}_B(b, a)$ and $\mathcal{R}(\langle b, a, b' \rangle, a') = 0$ for all $b, b' \in \mathcal{B}$ and $a, a' \in \mathcal{A}$. The density of the remaining transitions is null. We refer to the states of the form $\langle b, a, b' \rangle \in \mathcal{B} \times \mathcal{A} \times \mathcal{B}$ as *dummy states*.



Then, given any policy for $\mathcal{M}_B$, lifted to $\mathcal{M}'_B$ by executing any arbitrary action in dummy states of the form $\langle b, a, b' \rangle$, the expected return in $\mathcal{M}_B$ and $\mathcal{M}'_B$ is the same.

Proof. This follows by the construction of $\mathcal{M}'_B$. First, the discount factor is $\sqrt{\gamma}$. Second, the reward, as well as the transition functions are respectively null and deterministic in the dummy states. Third, notice that for every even time step, the agent is in a concrete belief state of $\mathcal{M}_B$, while at odd time steps, the agent is always in a dummy state. Then, we have for any policy π for $\mathcal{M}_B$ that

$$\mathbb{E}_{\pi}^{\mathcal{M}'_B} \left[\sum_{t=0}^{\infty} \gamma^{\frac{t}{2}} \mathcal{R}'_B(b_t^{\uparrow}, a_t) \right] = \mathbb{E}_{\pi}^{\mathcal{M}'_B} \left[\sum_{t=0}^{\infty} \gamma^{\frac{2t}{2}} \mathcal{R}'_B(b_{2t}^{\uparrow}, a_{2t}) \right] = \mathbb{E}_{\pi}^{\mathcal{M}_B} \left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_B(b_t, a_t) \right],$$

where we respectively denote by $b^{\uparrow}$ and b the beliefs coming from $\mathcal{B}'$ and $\mathcal{B}$. $\square$

Lifted latent policy. Let us now formally define the latent policy lifted in the belief MDP $\mathcal{M}'_B$ (the belief MDP which embeds dummy states, cf. Lemma 21). Given a latent stationary policy $\bar{\pi} : \bar{\mathcal{B}} \rightarrow \Delta(\bar{\mathcal{S}})$ for $\bar{\mathcal{M}}$ and the belief encoder Φ , we define the *lifted*

latent policy $\bar{\pi}^\star$ for $\mathcal{M}'_B$ as the Mealy machine $\bar{\pi}^\star = \langle \bar{B}, \bar{\pi}_a, \bar{\pi}_u, \bar{b}_I = \delta_{\bar{s}_I} \rangle$ (Property 1) with the following components: let $a \in \mathcal{A}$, $b^\dagger \in B'$, and $\bar{b}, \bar{b}' \in \bar{B}$,

(i) the next action function is defined as

$$\bar{\pi}_a(a \mid b^\dagger, \bar{b}) = \begin{cases} \bar{\pi}(a \mid \bar{b}) & \text{if } b^\dagger \text{ is a belief state, i.e., } b^\dagger \in B, \text{ and} \\ \mathbb{1}\{a = a'\} & \text{if } b^\dagger \text{ is a dummy state of the form } b^\dagger = \langle b, a', b' \rangle; \end{cases}$$

(ii) we define the density of the memory update function $\bar{\pi}_u$ as follows:

- $p_{\bar{\pi}_u}(\bar{b}' \mid b^\dagger, \bar{b}, a) = \delta_{\bar{b}}(\bar{b}')$ if $b^\dagger$ is *not* a dummy state, i.e., $b^\dagger \in B$,
- else, if $b^\dagger$ is a belief state of the form $b^\dagger = \langle b, a, b' \rangle$ with $b' \neq \delta_{s_{\text{reset}}}$,

$$p_{\bar{\pi}_u}(\bar{b}' \mid b^\dagger, \bar{b}, a) = \frac{\mathbb{E}_{s \sim b} \mathbb{E}_{s' \sim \mathcal{P}(\cdot \mid s, a)} \mathbb{E}_{o' \sim \mathcal{O}(\cdot \mid s', a)} [\delta_{\Phi(\bar{b}, a, o')}(\bar{b}') \cdot \delta_{\mathcal{T}(b, a, o')}(b')]}{\mathbb{E}_{s \sim b} \mathbb{E}_{s' \sim \mathcal{P}(\cdot \mid s, a)} \mathbb{E}_{o' \sim \mathcal{O}(\cdot \mid s', a)} \delta_{\mathcal{T}(b, a, o')}(b')},$$

and

- $\bar{\pi}_u(\bar{b}' \mid b^\dagger, \bar{b}, a) = \mathbb{1}\{\bar{b}' = \delta_{s_{\text{reset}}}\}$ otherwise.

Note that in every dummy state that does not embed a reset state, the belief update function is simply obtained by the usual conditional probability rule:

$$\Pr(\bar{b}' \mid b, \bar{b}, a, b') = \frac{\Pr(b', \bar{b}' \mid b, \bar{b}, a)}{\Pr(b' \mid b, \bar{b}, a)}$$

where $\Pr(\bar{b}' \mid b, \bar{b}, a, b') = p_{\bar{\pi}_u}(\bar{b}' \mid \langle b, a, b' \rangle, \bar{b}, a)$,

$$\Pr(b', \bar{b}' \mid b, \bar{b}, a) = \mathbb{E}_{s \sim b} \mathbb{E}_{s' \sim \mathcal{P}(\cdot \mid s, a)} \mathbb{E}_{o' \sim \mathcal{O}(\cdot \mid s', a)} [\delta_{\Phi(\bar{b}, a, o')}(\bar{b}') \cdot \delta_{\mathcal{T}(b, a, o')}(b')],$$

and $\Pr(b' \mid b, \bar{b}, a) = p_B(b' \mid b, a)$ (the next *original* belief state is independent from the current *latent* belief state given the current original belief).

Induced MCs and bisimulation. Let $\mathcal{M}'_{B, \bar{\pi}^\star} = \langle B' \times \bar{B}, P'_{B, \bar{\pi}^\star}, \mathcal{R}'_{B, \bar{\pi}^\star}, \langle b_I, \bar{b}_I \rangle, \sqrt{Y} \rangle$ be the Markov chain induced by the belief MDP of $\mathcal{P}$ (with dummy states) and the latent policy $\bar{\pi}^\star$ (cf. Definition 8). Let $b, b' \in B$, $\bar{b}, \bar{b}' \in \bar{B}$, and $a, a' \in \mathcal{A}$. The transition density function of $\mathcal{M}'_{B, \bar{\pi}^\star}$ is defined as follows:

$$p'_{B, \bar{\pi}^\star}(\langle b, a', b' \rangle, \bar{b} \mid b, \bar{b}) = \mathbb{E}_{a \sim \bar{\pi}(\cdot \mid \bar{b})} [p_B(b' \mid b, a) \cdot \delta_{a'}(a)] = \bar{\pi}(a' \mid \bar{b}) \cdot p_B(b' \mid b, a'), \text{ and}$$

$$p'_{B, \bar{\pi}^\star}(b', \bar{b}' \mid \langle b, a', b' \rangle, \bar{b}) = p_{\bar{\pi}_u}(\bar{b}' \mid \langle b, a', b' \rangle, \bar{b}, a').$$

The density of the remaining transitions is null. The reward function is merely obtained via its marginalization according to the action executed by $\bar{\pi}$:

$$\mathcal{R}'_{B,\bar{\pi}^\star}(b, \bar{b}) = \mathbb{E}_{a \sim \bar{\pi}(\cdot|\bar{b})} \mathcal{R}(b, a) \quad \text{and} \quad \mathcal{R}'_{B,\bar{\pi}^\star}(\langle b, a', b' \rangle, \bar{b}) = 0.$$

Let us now define the MC $\mathcal{M}_{B,\bar{\pi}^\star} = \langle \mathcal{B} \times \bar{\mathcal{B}}, \mathbf{P}_{B,\bar{\pi}^\star}, \mathcal{R}_{B,\bar{\pi}^\star}, \langle b_I, \bar{b}_I \rangle, \gamma \rangle$ as a copy of $\mathcal{M}'_{B,\bar{\pi}^\star}$ *without the dummy states*, defined over the state space $\mathcal{B} \times \bar{\mathcal{B}}$ (instead of $\mathcal{B}' \times \bar{\mathcal{B}}$) and with discount factor γ (instead of $\sqrt{\gamma}$). Precisely, we define the transition density of $\mathcal{M}_{B,\bar{\pi}^\star}$ as

$$\begin{aligned} \mathbf{p}_{B,\bar{\pi}^\star}(b', \bar{b}' | b, \bar{b}) &= \sum_{a \in \mathcal{A}} \mathbf{p}'_{B,\bar{\pi}^\star}(\langle b, a, b' \rangle, \bar{b} | b, \bar{b}) \cdot \mathbf{p}'_{B,\bar{\pi}^\star}(b', \bar{b}' | \langle b, a, b' \rangle, \bar{b}) \\ &= \mathbb{E}_{a \sim \bar{\pi}(\cdot|\bar{b})} [\mathbf{p}_B(b' | b, a) \cdot p_{\bar{\pi}_u}(\bar{b}' | \langle b, a, b' \rangle, \bar{b}, a)] \\ &= \mathbb{E}_{a \sim \bar{\pi}(\cdot|\bar{b})} \mathbb{E}_{s \sim b} \mathbb{E}_{s' \sim \mathbf{P}(\cdot|s, a)} \mathbb{E}_{o' \sim \mathcal{O}(\cdot|s', a)} [\delta_{\Phi(\bar{b}, a, o')}(b') \cdot \delta_{\mathcal{T}(b, a, o')}(b')] . \end{aligned}$$

Intuitively, $\mathbf{p}_{B,\bar{\pi}^\star}$ simply aggregates two consecutive steps from the function $\mathbf{p}'_{B,\bar{\pi}^\star}$ into a unique time step: (i) transitioning from $\langle b, \bar{b} \rangle$ to the next belief state b' through a dummy state $\langle b, a, b' \rangle$, and (ii) updating the latent belief to $\bar{b}'$ according to the action a recorded into the dummy state. The reward function is simply given by $\mathcal{R}_{B,\bar{\pi}^\star}(b, \bar{b}) = \mathcal{R}'_{B,\bar{\pi}^\star}(b, \bar{b})$. By construction (cf. Lemma 21), the MC $\mathcal{M}'_{B,\bar{\pi}^\star}$ is equivalent to $\mathcal{M}_{B,\bar{\pi}^\star}$, as the dummy states are only used to synchronize the Mealy machine with the next belief state: the expected return in both models is exactly the same.

Now, we show that the history unfolding $\mathcal{M}_H$ operating under $\bar{\pi}^\star$ is equivalent to the MC $\mathcal{M}_{B,\bar{\pi}^\star}$.

Lemma 22: Bisimulation between the history unfolding and the belief MDP

Denote by $\mathcal{M}_{H,\bar{\pi}^\star}$ the Markov chain induced by the history unfolding of $\mathcal{P}$ and the lifted policy $\bar{\pi}^\star$. There is a bisimulation relation between $\mathcal{M}_{H,\bar{\pi}^\star}$ and $\mathcal{M}_{B,\bar{\pi}^\star}$.

Proof. Define $\equiv_\Phi^r$ as the set $\{ \langle h, \langle b, \bar{b} \rangle \rangle \in S_H \times (\mathcal{B} \times \bar{\mathcal{B}}) \mid \mathcal{T}^*(h) = b \text{ and } \Phi^*(h) = \bar{b} \}$. We show that $\equiv_\Phi^r$ is a bisimulation relation between the states of $\mathcal{M}_{H,\bar{\pi}^\star}^r$ and $\mathcal{M}_{B,\bar{\pi}^\star}^r$. Let $h = \langle a_{0:T-1}, o_{1:T} \rangle \in S_H$, $b \in \mathcal{B}$, and $\bar{b} \in \bar{\mathcal{B}}$ so that $o_T \neq o_{\text{reset}}$ and $h \equiv_\Phi^r \langle b, \bar{b} \rangle$:^a

1. $\mathcal{R}_{H,\bar{\pi}^\star}(h) = \mathbb{E}_{a \sim \bar{\pi}(\cdot|\Phi^*(h))} \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathcal{R}(s, a) = \mathbb{E}_{a \sim \bar{\pi}(\cdot|\bar{b})} \mathbb{E}_{s \sim b} \mathcal{R}(s, a) = \mathcal{R}_{B,\bar{\pi}^\star}(b, \bar{b})$.
2. Note that histories are deterministically mapped to a *unique* belief (resp. latent belief) state through the belief update function $\mathcal{T}$ (resp. Φ). However, $\mathcal{T}$ and Φ are *not injective*, as multiple histories can lead to the same belief (resp. latent belief) state. Then, each equivalence class T is composed of belief and latent

belief pairs that aggregate multiple histories through $\mathcal{T}$ and Φ . Concretely, let $b' \in \mathcal{B}$, $\bar{b}' \in \bar{\mathcal{B}}$, an equivalence class of $\equiv_{\Phi}^{\tau}$ embeds subsets of the form $[\langle b', \bar{b}' \rangle]_{\equiv_{\Phi}^{\tau}}$, so that

(a) the intersection of $[\langle b', \bar{b}' \rangle]_{\equiv_{\Phi}^{\tau}}$ with $S_{\mathcal{H}}$ is the set

$$\{h' \in S_{\mathcal{H}} \mid \mathcal{T}^*(h') = b' \text{ and } \Phi^*(h') = \bar{b}'\}, \text{ and}$$

(b) the intersection of $[\langle b', \bar{b}' \rangle]_{\equiv_{\Phi}^{\tau}}$ with the state space of $\mathcal{M}_{\mathcal{B}}^{\bar{\pi}^*}$ merely consists of the pair $\langle b', \bar{b}' \rangle$.

Therefore,

$$\begin{aligned} & \int_{[\langle b', \bar{b}' \rangle]_{\equiv_{\Phi}^{\tau}}} p_{\mathcal{H}, \bar{\pi}^*}(h' \mid h) dh' \\ &= \int_{[\langle b', \bar{b}' \rangle]_{\equiv_{\Phi}^{\tau}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot \mid \Phi^*(h))} \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{s' \sim \mathcal{P}(\cdot \mid s, a)} \mathbb{E}_{o' \sim \mathcal{O}(\cdot \mid s', a)} \delta_{h-a-o'}(h') dh' \\ &= \int_{S_{\mathcal{H}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot \mid \Phi^*(h))} \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{s' \sim \mathcal{P}(\cdot \mid s, a)} \mathbb{E}_{o' \sim \mathcal{O}(\cdot \mid s', a)} \delta_{h-a-o'}(h') \cdot \delta_{\mathcal{T}^*(h')}(b') \cdot \delta_{\Phi^*(h')}(b') dh' \\ & \quad \text{(by definition of } [\langle b', \bar{b}' \rangle]_{\equiv_{\Phi}^{\tau}}) \\ &= \mathbb{E}_{a \sim \bar{\pi}(\cdot \mid \Phi^*(h))} \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{s' \sim \mathcal{P}(\cdot \mid s, a)} \mathbb{E}_{o' \sim \mathcal{O}(\cdot \mid s', a)} [\delta_{\mathcal{T}^*(h-a-o')}(b') \cdot \delta_{\Phi^*(h-a-o')}(b')] \\ &= \mathbb{E}_{a \sim \bar{\pi}(\cdot \mid \bar{b})} \mathbb{E}_{s \sim b} \mathbb{E}_{s' \sim \mathcal{P}(\cdot \mid s, a)} \mathbb{E}_{o' \sim \mathcal{O}(\cdot \mid s', a)} \delta_{\mathcal{T}(b, a, o')}(b') \cdot \delta_{\Phi(b, a, o')}(b') \quad (\text{since } h \equiv_{\Phi}^{\tau} \langle b, \bar{b} \rangle) \\ &= p_{\mathcal{B}, \bar{\pi}^*}(b', \bar{b}' \mid b, \bar{b}). \end{aligned}$$

By 1 and 2, we have that $\mathcal{M}_{\mathcal{H}}$ and $\mathcal{M}_{\mathcal{B}}$ are in bisimulation under the equivalence relation $\equiv_{\Phi}^{\tau}$, when the policies $\bar{\pi}^*$ and $\bar{\pi}^*$ are respectively executed in the two models. $\square$

^aThe case where $o_{\mathcal{T}} = o_{\text{reset}}$ is trivial.

Note that we only set up a bisimulation relation here, not necessarily the coarsest one. Since (i) any bisimulation relation between two models preserves the optimal expected return [Givan et al., 2003], and (ii) the two induced MCs are purely stochastic (there is no nondeterministic choice, so the expected return is unique in both models), we get the following corollary.

Corollary 6: The behaviors induced by $\bar{\pi}^\star$ in $\mathcal{M}_H$ and $\bar{\pi}^\star$ in $\mathcal{M}_B$ are the same

The agent behaviors formulated through the expected return that are obtained by executing the policies respectively in the two models are the same:
 $\mathbb{E}_{\bar{\pi}^\star}^{\mathcal{M}_H} [\mathcal{R}(s_I, a_0) + \sum_{t=1}^{\infty} \gamma^t \cdot \mathcal{R}_H(a_{0:t-1}, o_{1:t})] = \mathbb{E}_{\bar{\pi}^\star}^{\mathcal{M}_B} [\sum_{t=0}^{\infty} \gamma^t \cdot \mathcal{R}_B(b_t, a_t)].$

Concluding the proof of Proposition 1. In consequence, since $\mathcal{M}_H$ and $\mathcal{M}_B$ are equivalent when the latent policy $\bar{\pi}$ is executed via the belief encoder Φ and induce the same behaviors, it is sufficient to consider $\mathcal{M}_{H, \bar{\pi}^\star}$ to study the behaviors of the agent in the POMDP $\mathcal{P}$ when the latent policy is executed. Since $\mathcal{M}_H$ is episodic, the existence of a stationary distribution is guaranteed by Theorem 7.

Theorem 29: Unique stationary distribution over the histories of $\mathcal{P}$

There exists a unique stationary distribution $\mathcal{H}_{\bar{\pi}} \in \Delta((\mathcal{A} \cdot \Omega)^*)$ over the states of the history unfolding $\mathcal{M}_{H, \bar{\pi}^\star}$.

2.3 Local Losses

The objective function of the WAE-MDP incorporates *local losses* (Chapter 3, Section 1.1) that minimize the expected distance between the original and latent reward and transition functions. By taking into account the fact that the dynamics are learned from the refined MDP $\mathcal{M}_\Omega$, the local losses are given by

$$L_R = \mathbb{E}_{s, o, a \sim \mathcal{H}_{\bar{\pi}}} |\mathcal{R}(s, a) - \bar{\mathcal{R}}(\phi(s, o), a)| \quad \text{and} \quad L_P = \mathbb{E}_{s, o, a \sim \mathcal{H}_{\bar{\pi}}} \mathcal{W}_d(\phi \mathbf{P}_\Omega(\cdot \mid s, o, a), \bar{\mathbf{P}}(\cdot \mid \phi(s, o), a)),$$

where $s, o, a \sim \mathcal{H}_{\bar{\pi}}$ is a shorthand for drawing (i) $h \sim \mathcal{H}_{\bar{\pi}}$ so that o is the last observation of h , (ii) $s \sim \mathcal{T}^*(h)$, and (iii) $a \sim \bar{\pi}(\cdot \mid \Phi^*(h))$. In practice, *the ability of observing the states during learning enables the optimization of the local losses without the need of explicitly storing histories*. Instead, we simply store the transitions of $\mathcal{M}_\Omega$ encountered while executing $\bar{\pi}$ into a replay buffer. We also introduce an *observation loss* in addition to the reconstruction loss of the decoder, which allows learning $\bar{\mathcal{O}}$:

$$L_{\mathcal{O}} = \mathbb{E}_{s, o, a \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{s' \sim \mathbf{P}(\cdot \mid s, a)} d_{TV} \left(\mathcal{O}(\cdot \mid s', a), \mathbb{E}_{o' \sim \bar{\mathcal{O}}(\cdot \mid s', a)} \bar{\mathcal{O}}(\cdot \mid \phi(s', o')) \right). \quad (4.5)$$

Belief losses. In Equation 4.4, we set D as the Wasserstein distance between the true latent belief update and the belief encoder. In addition, we introduce the following *reward and transition regularizers* to reconcile the behaviors obtained in the fully observable

latent model $\bar{\mathcal{M}}$ and the partially observable one $\bar{\mathcal{P}}$:

$$L_{\bar{\mathcal{T}}} = \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} \mathcal{W}_{\bar{\mathcal{d}}}(\bar{\mathcal{T}}^*(h), \Phi^*(h)), \quad L_{\bar{\mathcal{R}}}^{\Phi} = \mathbb{E}_{h, s, o, a \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{\bar{s} \sim \Phi^*(h)} |\bar{\mathcal{R}}(\phi(s, o), a) - \bar{\mathcal{R}}(\bar{s}, a)|, \text{ and}$$

$$L_{\bar{\mathcal{P}}}^{\Phi} = \mathbb{E}_{h, s, o, a \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{\bar{s} \sim \Phi^*(h)} \mathcal{W}_{\bar{\mathcal{d}}}(\bar{\mathcal{P}}(\cdot | \phi(s, o), a), \bar{\mathcal{P}}(\cdot | \bar{s}, a)). \quad (4.6)$$

The losses $L_{\bar{\mathcal{R}}}^{\Phi}$ and $L_{\bar{\mathcal{P}}}^{\Phi}$ aim at regularizing Φ and minimize the difference between the rewards (resp. transition probabilities) that are expected when drawing states from the current belief compared to those actually observed. Again, the ability to observe the states during training enables optimizing the losses without explicitly requiring the states to execute the policy. The belief loss and the related two regularizers can be optimized *on-policy*, i.e., coupled with the optimization of the latent policy $\bar{\pi}$ that is used to generate the episodes.

To help the reader grasp the intuition behind those loss functions, we provide a high-level overview of the optimization process in Figure 4.5.

The procedure allowing to minimize the reward, transition, and observation losses is depicted in Figure 4.5a. At each step, a state s , an observation o of s , and an action a are drawn from the distribution of experiences encountered $\mathcal{H}_{\bar{\pi}}$ while executing $\bar{\pi}$. First, $\langle s, o \rangle$ is mapped to the latent space via the state embedding function of the WAE-MDP, i.e., $\phi(s, o) = \bar{s}$. Then, the action a is executed both in the original and latent space models (respectively from $\langle s, o \rangle$ and $\bar{s}$), which allows quantifying the distance between the next reward and transition produced in the original and latent models. Finally, the original model transitions to the next state-observation pair $\langle s', o' \rangle$. Embedding this pair again to the latent space via $\phi(s', o') = \bar{s}'$ allows to quantify the distance between the original observation o' and the one that is produced in the latent space model, from $\bar{s}'$ via $\bar{o}' \sim \bar{\mathcal{O}}(\cdot | \bar{s}')$.

The procedure allowing to minimize the belief losses is depicted in Figure 4.5b. This time, the optimization is performed *on-policy*, which means that it is performed while executing the policy in the original environment. At time step $t \geq 1$, the current history h ends up in the observation o_t of state s_t . First, the discrepancy between the latent belief obtained via our belief encoder $\bar{b}_t = \Phi^*(h)$ and the one obtained via the true latent belief update function $\bar{b}' = \bar{\mathcal{T}}^*(h)$ is evaluated and can further be minimized from that point. Second, we compute the reward and transition regularizers by minimizing the distance between rewards and transitions produced from believed states $\bar{s}_t \sim \bar{b}_t$ (i.e., states witnessed by the current belief $\bar{b}_t$) and those produced by mapping the current state-observation pair into the latent space, via $\phi(s_t, o_t) = \bar{s}$, when the action $a \sim \bar{\pi}(\cdot | \bar{b}_t)$ is produced. Finally, a is executed in the original environment and the process is repeated until the end of the episode.

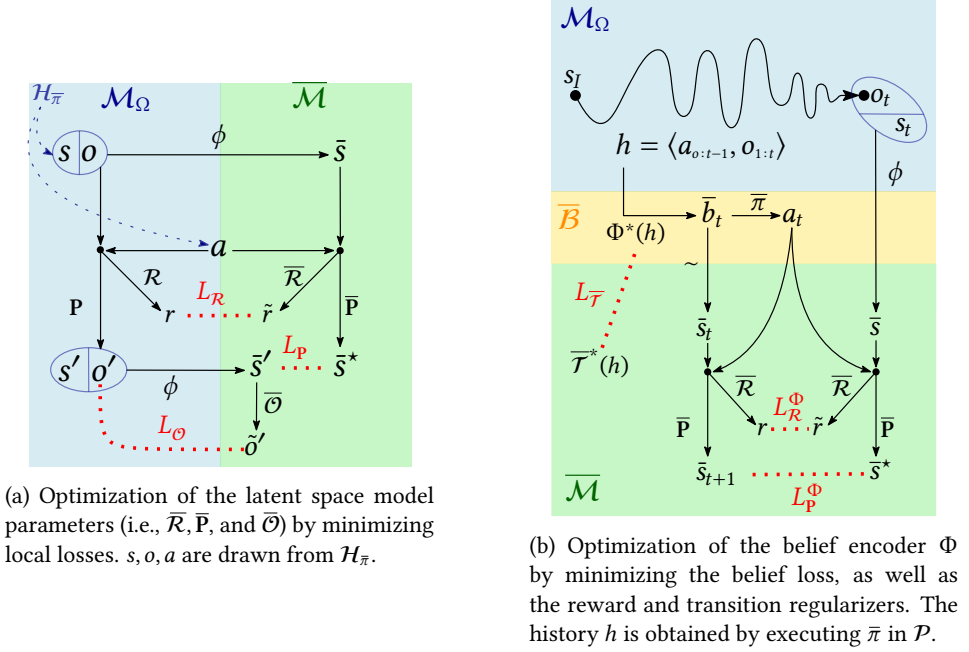


Figure 4.5: Local losses. Arrows represent (stochastic) mappings, the original state-observation (resp. latent state) space is spread along the blue (resp. green) area, and the latent belief space is spread along the yellow area. Distances are depicted in red. Notice that the blue area corresponds to the state-observation space $\mathcal{S}_{\Omega}$, which is observable only during training (Assumption 9).

2.4 Value Difference Bounds

We provide guarantees concerning the *agent behaviors in $\mathcal{P}$* , when the policies are *conditioned on latent beliefs inferred from Φ* . We formalize the behaviors of the agent through the expected returns achieved in both models under the latent policy. We claim that when the losses are minimized to zero,

- (i) (*Model quality*): the latent model almost surely mimics the original environment, and
- (ii) (*Representation quality*): our belief representation captures the value function; in other words, the closeness of the representation of *consistent* histories in the latent space implies the closeness of their values.

Consistent means that we range over histories that are measurable under both $\mathcal{P}$ and $\bar{\mathcal{P}}$. Precisely, let

$$\mathcal{L} = L_{\mathcal{R}} + L_{\bar{\mathcal{R}}}^{\Phi} + R_{\max} \cdot L_{\bar{\mathcal{T}}} + \gamma K_{\bar{V}} \cdot (L_{\mathcal{P}} + L_{\bar{\mathcal{P}}}^{\Phi} + L_{\bar{\mathcal{T}}} + L_{\mathcal{O}}),$$

with $K_{\bar{V}} = R_{\max}/(1-\gamma)$. Then, we have the following results.

Theorem 30: Model quality

For any latent policy $\bar{\pi} : \bar{\mathcal{B}} \rightarrow \Delta(\mathcal{A})$, the values of $\mathcal{P}$ and $\bar{\mathcal{P}}$ are bounded by the local and belief losses in average when, in both models, the actions are produced by $\bar{\pi}$, which is conditioned on the latent belief induced by Φ , i.e., $a \sim \bar{\pi}(\cdot \mid \Phi^*(h))$:

$$\mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} |V^{\bar{\pi}}(h) - \bar{V}^{\bar{\pi}}(h)| \leq \frac{\mathcal{L}}{1-\gamma}. \quad (4.7)$$

Proof sketch. Given the intricate and computational nature of the proof, we postpone its detailed exposition to Appendix C.1. Nevertheless, we give some ideas of the calculations required to infer the bound:

- (i) we develop the value difference inside the absolute value by exploiting the fact that the value function is the unique solution of Bellman's equation:

$$\begin{aligned} & \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} |V^{\bar{\pi}}(h) - \bar{V}^{\bar{\pi}}(h)| \\ &= \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{a \sim \bar{\pi}(\cdot \mid \Phi^*(h))} \left[\mathcal{R}(s, a) + \gamma \cdot \mathbb{E}_{s' \sim \mathcal{P}(\cdot \mid s, a)} \mathbb{E}_{o' \sim \mathcal{O}(\cdot \mid s', a)} V^{\bar{\pi}}(h \cdot a \cdot o') \right] \right. \\ & \quad \left. - \mathbb{E}_{\bar{s} \sim \bar{\mathcal{T}}^*(h)} \mathbb{E}_{a \sim \bar{\pi}(\cdot \mid \Phi^*(h))} \left[\bar{\mathcal{R}}(\bar{s}, a) + \gamma \cdot \mathbb{E}_{\bar{s}' \sim \bar{\mathcal{P}}(\cdot \mid \bar{s}, a)} \mathbb{E}_{o' \sim \bar{\mathcal{O}}(\cdot \mid \bar{s}', a)} \bar{V}^{\bar{\pi}}(h \cdot a \cdot o') \right] \right|; \end{aligned}$$

- (ii) we repeatedly apply the triangle and the Jensen's inequalities to reveal mean discrepancies for either rewards, value functions, and observations;
- (iii) we bound those discrepancies by L_1 norms and Wasserstein terms, which reveal the local losses;
- (iv) when all the local losses from $\mathcal{L}$ are revealed, we are left with a discounted value difference term over the *next* possible histories. Then, we exploit the stationary property of $\mathcal{H}_{\bar{\pi}}$ to retrieve a discounted, absolute value difference term over the *current* history:

$$\mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} |V^{\bar{\pi}}(h) - \bar{V}^{\bar{\pi}}(h)|$$

$$\begin{aligned}
 &\leq \mathcal{L} + \gamma \cdot \mathbb{E}_{h, o \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{s', o' \sim \mathcal{P}_{\Omega}(\cdot | s, o, a)} \left| V^{\bar{\pi}}(h \cdot a \cdot o') - \bar{V}^{\bar{\pi}}(h \cdot a \cdot o') \right| \\
 &= \mathcal{L} + \gamma \cdot \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} |V^{\bar{\pi}}(h) - \bar{V}^{\bar{\pi}}(h)|.
 \end{aligned}$$

Solving the equation yields the $1/(1-\gamma)$ term.

- (v) Putting all together, we derive an inequality only composed of constants, multiplied by the losses that we minimize.

□

Theorem 31: Representation quality

Let $\bar{\pi}$ be any latent policy conditioned on latent beliefs inferred from Φ . Then for any $n \geq 0$, there exists a $K \geq 0$ so that for any histories h_1, h_2 of length at most n that are measurable under both $\mathcal{P}$ and $\bar{\mathcal{P}}$ with $\Phi^*(h_1) = \bar{b}_1$ and $\Phi^*(h_2) = \bar{b}_2$, the representation induced by Φ yields:

$$|V^{\bar{\pi}}(h_1) - V^{\bar{\pi}}(h_2)| \leq \frac{K}{2} \|\bar{b}_1 - \bar{b}_2\|_1 + \frac{K \cdot L_{\bar{\pi}} + \mathcal{L}}{1 - \gamma} \left(\frac{1}{\mathcal{H}_{\bar{\pi}}(h_1)} + \frac{1}{\mathcal{H}_{\bar{\pi}}(h_2)} \right). \quad (4.8)$$

Intuitively, the theorem states that the latent representation captures the value function: whenever the losses are minimized and go to zero, *histories that are encoded to close representations are guaranteed to have close values as well*. Phrased differently, Theorem 31 ensures that the representation induced by our encoder is suitable to optimize the (latent) policy since the distance between beliefs in the latent space characterizes the distance of values in the original environment. The latent belief space thus captures the necessary information to learn a policy that optimizes the expected return.

Proof (of Theorem 31). Let $n \geq 0$ and h_1, h_2 be two measurable histories so that $h_i = \langle a_0 : T_i - 1, o_1 : T_i \rangle$ with $T_i \leq n$ and $o_t \in \bigcup_{\bar{s} \in \bar{\mathcal{S}}} \text{supp}(\bar{\mathcal{O}}(\cdot | \bar{s}))$ for all $t < T_i$ and $i \in \{1, 2\}$ (the two histories are consistent with both $\mathcal{P}$ and $\bar{\mathcal{P}}$). Then, we have:

$$\begin{aligned}
 &|V^{\bar{\pi}}(h_1) - V^{\bar{\pi}}(h_2)| \\
 &= |V^{\bar{\pi}}(h_1) - \bar{V}^{\bar{\pi}^*}(h_1) + \bar{V}^{\bar{\pi}^*}(h_1) - \bar{V}^{\bar{\pi}^*}(h_2) + \bar{V}^{\bar{\pi}}(h_2) - V^{\bar{\pi}}(h_2)| \\
 &\leq |V^{\bar{\pi}}(h_1) - \bar{V}^{\bar{\pi}}(h_1)| + |\bar{V}^{\bar{\pi}}(h_1) - \bar{V}^{\bar{\pi}}(h_2)| + |\bar{V}^{\bar{\pi}}(h_2) - V^{\bar{\pi}}(h_2)| \quad (\text{Triangle inequality}) \\
 &\leq \mathcal{H}_{\bar{\pi}}(h_1)^{-1} \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} |V^{\bar{\pi}}(h) - \bar{V}^{\bar{\pi}}(h)| + |\bar{V}^{\bar{\pi}}(h_1) - \bar{V}^{\bar{\pi}}(h_2)| \\
 &\quad + \mathcal{H}_{\bar{\pi}}(h_2)^{-1} \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} |V^{\bar{\pi}}(h) - \bar{V}^{\bar{\pi}}(h)| \\
 &\hspace{15em} (\text{Lemma 13})
 \end{aligned}$$

$$\leq |\bar{V}^{\bar{\pi}}(h_1) - \bar{V}^{\bar{\pi}}(h_2)| + \frac{\mathcal{L}}{1-\gamma} (\mathcal{H}_{\bar{\pi}}(h_1)^{-1} + \mathcal{H}_{\bar{\pi}}(h_2)^{-1}) \quad (\text{Theorem 30})$$

First, notice that $\bar{\mathcal{P}}$ is finite, i.e., $\bar{\mathcal{P}}$ is defined over finite spaces. To see why, note that the latent state space $\bar{\mathcal{S}}$ is finite. Then, $\bar{\mathcal{O}}$ is deterministic by definition (it consists of the Dirac centered on the observation produced by $\bar{\mathcal{O}}_\mu$), so the observations of $\bar{\mathcal{P}}$ can be limited to the union of the supports of $\bar{\mathcal{O}}(\cdot | \bar{s})$, ranging over all latent states $\bar{s} \in \bar{\mathcal{S}}$. The resulting set is also finite.^a

Second, notice that we consider histories from the set

$$\text{Hist}_n = \{h \in (\mathcal{A} \cdot \Omega)^* : \{h\} \in \text{supp}(\mathcal{H}_{\bar{\pi}^*}), h \in \mathbf{H}(\mathcal{P}, \bar{\mathcal{P}}), \text{ and } |h| \leq n\},$$

where $\text{supp}(\mathcal{H}_{\bar{\pi}^*})$ denotes the support of $\mathcal{H}_{\bar{\pi}^*}$, $|h|$ the length of the history h , and $\mathbf{H}(\mathcal{P}, \bar{\mathcal{P}})$ the set of histories consistent with both $\mathcal{P}$ and $\bar{\mathcal{P}}$, i.e., histories whose observations belong to $\bigcup_{\bar{s} \in \bar{\mathcal{S}}} \text{supp}(\bar{\mathcal{O}}(\cdot | \bar{s}))$. Notice that Hist_n is finite: we consider histories that are measurable in the two models (i.e., the original and latent POMDPs) and the observation space of the latent POMDP is finite. So, Hist_n consists of a finite number of action-observation branchings. In consequence, the set of reachable latent beliefs

$$\text{Beliefs}_n^\Phi = \{\bar{b} \in \Delta(\bar{\mathcal{S}}) \mid \text{there is a history } h \in \text{Hist}_n \text{ so that } \bar{b} = \Phi^*(h)\}$$

is finite as well. We define $\text{Beliefs}_n^{\bar{\mathcal{T}}}$ analogously. With the same reasoning, we can show that $\text{Beliefs}_n^{\bar{\mathcal{T}}}$ is finite.

Then, let the constant

$$K = \max_{\substack{\bar{b}_1, \bar{b}_2 \in \text{Beliefs}_n^\Phi \cup \text{Beliefs}_n^{\bar{\mathcal{T}}} \\ \bar{b}_1 \neq \bar{b}_2}} \frac{2|\bar{V}^{\bar{\pi}}(\bar{b}_1) - \bar{V}^{\bar{\pi}}(\bar{b}_2)|}{\|\bar{b}_1 - \bar{b}_2\|_1}. \quad (4.9)$$

We have:

$$\begin{aligned} & |\bar{V}^{\bar{\pi}}(h_1) - \bar{V}^{\bar{\pi}}(h_2)| \\ &= |\bar{V}^{\bar{\pi}}(\bar{\mathcal{T}}^*(h_1)) - \bar{V}^{\bar{\pi}}(\bar{\mathcal{T}}^*(h_2))| \\ &= |\bar{V}^{\bar{\pi}}(\bar{\mathcal{T}}^*(h_1)) - \bar{V}^{\bar{\pi}}(\Phi^*(h_1)) + \bar{V}^{\bar{\pi}}(\Phi^*(h_1)) - \bar{V}^{\bar{\pi}}(\Phi^*(h_2)) + \bar{V}^{\bar{\pi}}(\Phi^*(h_2)) - \bar{V}^{\bar{\pi}}(\bar{\mathcal{T}}^*(h_2))| \\ &\leq |\bar{V}^{\bar{\pi}}(\bar{\mathcal{T}}^*(h_1)) - \bar{V}^{\bar{\pi}}(\Phi^*(h_1))| + |\bar{V}^{\bar{\pi}}(\Phi^*(h_1)) - \bar{V}^{\bar{\pi}}(\Phi^*(h_2))| + |\bar{V}^{\bar{\pi}}(\Phi^*(h_2)) - \bar{V}^{\bar{\pi}}(\bar{\mathcal{T}}^*(h_2))| \\ &\quad (\text{Triangle inequality}) \\ &\leq K \cdot \frac{1}{2} \|\bar{\mathcal{T}}^*(h_1) - \Phi^*(h_1)\|_1 + K \cdot \frac{1}{2} \|\bar{b}_1 - \bar{b}_2\|_1 + K \cdot \frac{1}{2} \|\bar{\mathcal{T}}^*(h_2) - \Phi^*(h_2)\|_1 \end{aligned} \quad (\text{Equation 4.9})$$

$$\begin{aligned}
&= K \cdot \mathcal{W}_{\bar{d}}(\bar{\mathcal{T}}^*(h_1), \Phi^*(h_1)) + K \cdot \frac{1}{2} \|\bar{b}_1 - \bar{b}_2\|_1 + K \cdot \mathcal{W}_{\bar{d}}(\bar{\mathcal{T}}^*(h_2), \Phi^*(h_2)) \\
&\quad \text{(By definition of total variation, Definition 34)} \\
&\leq \mathcal{H}_{\bar{\pi}}(h_1)^{-1} \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} [K \cdot \mathcal{W}_{\bar{d}}(\bar{\mathcal{T}}^*(h), \Phi^*(h))] + K \cdot \frac{1}{2} \|\bar{b}_1 - \bar{b}_2\|_1 \\
&\quad + \mathcal{H}_{\bar{\pi}}(h_2)^{-1} \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} [K \cdot \mathcal{W}_{\bar{d}}(\bar{\mathcal{T}}^*(h), \Phi^*(h))] \\
&\quad \text{(Lemma 13)} \\
&= (\mathcal{H}_{\bar{\pi}}(h_1)^{-1} + \mathcal{H}_{\bar{\pi}}(h_2)^{-1}) \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} [K \cdot \mathcal{W}_{\bar{d}}(\bar{\mathcal{T}}^*(h), \Phi^*(h))] + K \cdot \frac{1}{2} \|\bar{b}_1 - \bar{b}_2\|_1 \\
&= (\mathcal{H}_{\bar{\pi}}(h_1)^{-1} + \mathcal{H}_{\bar{\pi}}(h_2)^{-1}) (K \cdot L_{\bar{\mathcal{T}}}) + K \cdot \frac{1}{2} \|\bar{b}_1 - \bar{b}_2\|_1. \quad \text{(by definition of } L_{\bar{\mathcal{T}}})
\end{aligned}$$

Putting all together, we have that for any $n \geq 0$, one can find a constant $K \geq 0$ so that:

$$|V^{\bar{\pi}}(h_1) - V^{\bar{\pi}}(h_2)| \leq \frac{K}{2} \|\bar{b}_1 - \bar{b}_2\|_1 + \frac{KL_{\bar{\mathcal{T}}} + \mathcal{L}}{1 - \gamma} (\mathcal{H}_{\bar{\pi}}(h_1)^{-1} + \mathcal{H}_{\bar{\pi}}(h_2)^{-1}).$$

□

^aNote that the same reasoning holds for any latent observation function with finite support.

Remark 27: Comparison to the representation guarantees of Avalos et al. [2024]

The representation guarantees of Avalos et al. [2024] do only hold for the latent policies $\bar{\pi}^*$ that are executed by processing histories via the belief encoder Φ and are optimal in the latent POMDP $\bar{\mathcal{P}}$. In practice, although optimal policies are a yardstick for evaluating the agent's performance, they are rarely the ones that are eventually learned, and reaching them during the learning process is often hardly feasible. In contrast, our result holds for *any* latent policy (therefore including such optimal ones).

The key difference between the two results lies in the way the constant K is set. The proof of Avalos et al. [2024] relies on the fact that the *optimal* value function of the latent POMDP is *almost Lipschitz continuous* in the latent belief space [Vanderbei, 1991] (for any ϵ , one can find a constant K so that $|\bar{V}^{\bar{\pi}^*}(\bar{b}_1) - \bar{V}^{\bar{\pi}^*}(\bar{b}_2)| \leq K \cdot d_{TV}(\bar{b}_1, \bar{b}_2) + \epsilon$), and then use that result in the derivations performed from Equation 4.10. Precisely, the almost Lipschitz continuity comes from the fact that the optimal value function of a model with finite spaces is *piecewise linear convex* (PWLC) when the horizon is fixed [Smallwood and Sondik, 1973; Sondik, 1978; Hauskrecht, 2000]. Then, one simply needs to find a horizon large enough to

ensure an error of at most ϵ with respect to the optimal (infinite horizon) value function. Finally, it suffices to take K as the higher slope of those linear functions that compose the PWLC function.

In our case, the proof relies on the fact that the set of *reachable latent beliefs* is finite since the length of the histories that we consider is bounded. The constant K can be obtained by taking the maximum ratio between the value difference and the distance of beliefs in the latent space, where we range over all *reachable* latent beliefs (see Equation 4.9). The resulting constant holds for all latent policies.

3 Learning to Believe

As we already hinted in the introduction of this chapter, the latent POMDP, policy, and belief updater are learned in a round-robin fashion (Figure 4.3). The latent POMDP $\bar{\mathcal{P}}$ is learned via the WAE-MDP framework by minimizing the local losses L_P , L_R , and L_O . The latent policy $\bar{\pi}$ is learned via an on-policy algorithm (in this case, A2C, cf. Algorithm 4), and the latent policy is conditioned on the latent belief inferred from the belief encoder Φ , which is updated by minimizing the belief losses $L_{\bar{\pi}}$, $L_{\bar{R}}^{\Phi}$, and $L_{\bar{P}}^{\Phi}$, over the trajectories produced by the latent policy in the original environment $\mathcal{P}$.

In this section, we specifically focus on the learning of the belief encoder Φ and provide details on the optimization procedure.

Architecture. Our latent belief encoder Φ aims at generalizing to *any* POMDP. Therefore, we do not make any assumption about the underlying belief distribution. To accommodate complex belief distributions, we use a *masked autoregressive flows* (MAF, Papamakarios et al., 2017, cf. Definition 42). Precisely, to fit with the WAE-MDP framework and leverage the guarantees presented in Section 2.4, we use the MAF described in Chapter 3, Section 3.4. We refer to the vector which embeds the parameters of the belief distribution

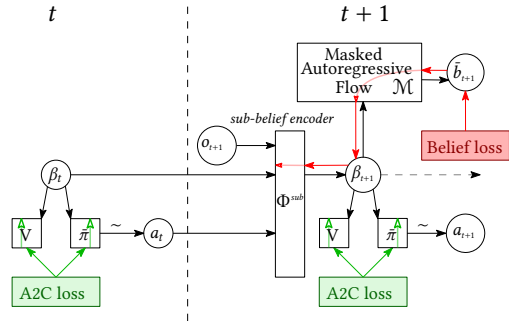


Figure 4.6: WBU learns to encode the history into a *sub-belief* solely by optimizing the belief loss. The policy, being conditioned on the sub-belief, is learned via A2C and does not back-propagate through the sub-belief encoder. The colored arrows represent the gradient flows of the different losses.

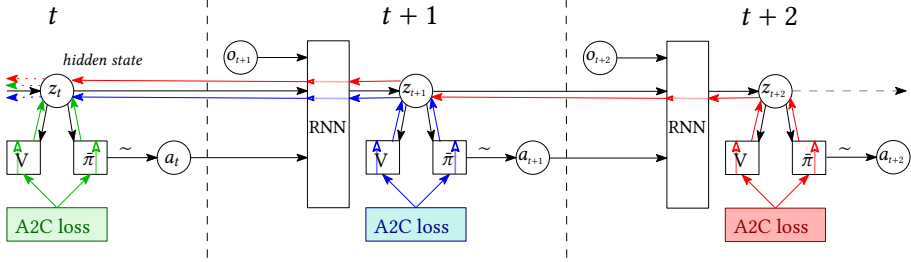


Figure 4.7: R-A2C combines A2C with RNNs to process the history at each time step $t \geq 0$, and the agent’s policy is conditioned on the hidden state z_t of the RNN. To optimize the parameters of the network, R-A2C uses backpropagation through time (BPTT): the RNN leverages gradients from future time steps to improve its compression of the history. The colored arrows represent the gradient flows of the different losses.

at each time step $t \geq 0$ as the *sub-belief* β_t , which is converted into a belief via the MAF $\mathcal{M}(\beta_t) = \bar{b}_t$. We use a *sub-belief encoder* Φ^{sub} to recursively update β_t via $\Phi^{\text{sub}}(\beta_t, a_t, o_{t+1}) = \beta_{t+1}$, so that $\Phi(\bar{b}_t, a_t, o_{t+1}) = \mathcal{M} \circ \Phi^{\text{sub}}(\beta_t, a_t, o_{t+1})$. RNNs are trained via backpropagation through time (BPTT), which may lead to learning instabilities as it accumulates errors over time, is computationally intensive, and requires mitigating exploding and vanishing gradients [Pascanu et al., 2013]. In contrast, albeit sub-beliefs are updated recursively in the same spirit as RNN hidden states, we do not need to use BPTT and use a simple feed-forward network for Φ^{sub} , as illustrated in Figure 4.6.

In R-A2C (Figure 4.7), the RNN hidden states serve as a compact representation of histories for the policy, which is learned through a *backward recurrence*: the errors in the value approximation are accumulated from the end to the beginning of the episode, meaning that the values of time steps closer to the end of an episode are easier to learn and the gradients of future time steps tend to be more accurate. Thus, BPTT helps learning.

This is in stark contrast to learning the belief update rule, which is learned through a *forward recurrence*: the errors in the belief approximation are accumulated from the beginning to the end of the episodes, which means that beliefs of early time steps are easier to infer. So BPTT is unnecessary and might even be harmful.

Additionally, viewing the belief update function as the transition function of $\mathcal{M}_B$, disabling BPTT aligns with the literature on model-based RL for learning Markovian transition functions (see, e.g., Gelada et al., 2019; François-Lavet et al., 2019 or V- and WAE-MDPs in Chapter 3, Sections 2 and 3).

In practice, we stop the gradients from previous time steps to avoid the accumulation of errors in the gradients.

Training. Our goal is to train Φ^{sub} and $\mathcal{M}$ to approximate the latent belief update rule, by minimizing the Wasserstein distance between the belief update function $\bar{\mathcal{T}}$ of the latent POMDP, and the belief encoder Φ (Equation 4.6), to leverage the theoretical learning guarantees of Theorem 30 and 31. However, Wasserstein optimization is challenging, often requiring the use of additional networks, Lipschitz constraints, and a min-max optimization procedure [Arjovsky et al., 2017; Gulrajani et al., 2017], similar to how WAE-MDPs are trained (Chapter 3, Section 3). Importantly, sampling from both distributions ($\bar{\mathcal{T}}$ and Φ) is necessary for optimizing the Wasserstein distance and, while sampling from our belief approximation is straightforward, sampling from the update rule (Equation 4.3) is non-trivial.

In practice, as an alternative to the Wasserstein optimization, we minimize the KL divergence between the two distributions (Definition 32). While the D_{KL} proxy is easier to optimize and only requires sampling from one of the two distributions (in our case, the belief encoder), it bounds the Wasserstein distance by the Pinsker’s inequality (Theorem 18) in the zero-temperature limit of the WAE-MDP.

On-policy KL divergence. Using D_{KL} as a proxy for $\mathcal{W}_{\bar{d}}$ allows to narrow the discrepancy between Φ and $\bar{\mathcal{T}}$. We train Φ *on-policy*, with the same samples as those used for $\bar{\pi}$, which helps to learn despite gradients not being allowed to flow between the networks. At any time-step $t \geq 0$, given the current belief $\bar{b}_t$, the action a_t played by the agent, and the next perceived observation o_{t+1} , the belief proxy loss is given by

$$D_{\text{KL}}(\Phi(\bar{b}_t, a_t, o_{t+1}) \parallel \bar{\mathcal{T}}(\bar{b}_t, a_t, o_{t+1})) = \log \left(\mathbb{E}_{\bar{s} \sim \bar{b}_t} \mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot | \bar{s}, a_t)} p_{\bar{\mathcal{O}}} (o_{t+1} \mid \bar{s}') \right) + \mathbb{E}_{\bar{s}_{t+1} \sim \Phi(\bar{b}_t, a_t, o_{t+1})} \left[\log p_{\Phi}(\bar{s}_{t+1} \mid \bar{b}_t, a_t, o_{t+1}) - \log \mathbb{E}_{\bar{s} \sim \bar{b}_t} \bar{\mathbf{p}}(\bar{s}_{t+1} \mid \bar{s}, a_t) - \log p_{\bar{\mathcal{O}}}(o_{t+1} \mid \bar{s}_{t+1}) \right], \quad (4.11)$$

where $p_{\bar{\mathcal{O}}}$, p_{Φ} , and $\bar{\mathbf{p}}$ respectively denote the densities of the latent observation function $\bar{\mathcal{O}}$, the belief encoder Φ , and the latent transition function $\bar{\mathbf{P}}$. Equation 4.11 consists of four terms: a normalization factor, the negative entropy of Φ , the alignment of the latent belief update with the transition function of the latent MDP, and the filtration of latent states unrelated to o_{t+1} .

Given the parameters ϑ of the network modeling Φ , we minimize the KL divergence D_{KL} by gradient descent on the Monte-Carlo estimate of the divergence:

$$\nabla_{\vartheta} D_{\text{KL}}(\Phi(\bar{b}_t, a_t, o_{t+1}; \vartheta) \parallel \bar{\mathcal{T}}(\bar{b}_t, a_t, o_{t+1})) = \nabla_{\vartheta} \mathbb{E}_{\bar{s}_{t+1} \sim \Phi(\bar{b}_t, a_t, o_{t+1}; \vartheta)} \left[\log p_{\Phi}(\bar{s}_{t+1} \mid \bar{b}_t, a_t, o_{t+1}; \vartheta) - \log \mathbb{E}_{\bar{s} \sim \bar{b}_t} \bar{\mathbf{p}}(\bar{s}_{t+1} \mid \bar{s}, a_t) - \log p_{\bar{\mathcal{O}}}(o_{t+1} \mid \bar{s}_{t+1}) \right]$$

Notice that the first term of the divergence (the belief normalization term) of Equation 4.11 does not depend on Φ and thus vanishes. Nevertheless, we observed during our experiments that adding the normalizing term allows for stabilizing and reducing the variance of the belief loss.

To learn to minimize this divergence through gradient descent, we use

- (i) a normal distribution for modeling the latent observation function $p_{\bar{\mathcal{O}}}$, which allows retrieving a Dirac as the variance of the normal goes to zero (Remark 25); and
- (ii) continuous relaxation of discrete distributions (see Chapter 3, Section 2.4) to model p_{Φ} and $\bar{\mathbf{p}}$ (more specifically, the MAF of Chapter 3, Section 3.4), which allows to retrieve discrete distributions as the temperature of the relaxed distributions go to zero (see Remarks 21 and 23).

Remark 28: Smoothness of latent observations

The WAE-MDP learns from the augmented POMDP $\mathcal{P}^{\dagger}$ (Section 2.1), which is equivalent to $\mathcal{P}$ and possesses a deterministic observation function. Such a deterministic mapping is learned in the WAE-MDP through $\bar{\mathcal{O}}_{\mu}$. However, when deterministic, the observation terms of Equations 4.3 and 4.11 are Dirac, which prevents learning: deterministically filtering out states from the next belief that does not share the next observation is sample inefficient and may further require constructing the full belief distribution, which is usually intractable. In practice, to alleviate this, we model the observation function as a normal distribution: $\bar{\mathcal{O}}(\cdot \mid \bar{s}) = \mathcal{N}(\cdot; \bar{\mathcal{O}}_{\mu}(\bar{s}), \sigma^2)$ where the variance can, e.g., be learned via $L_{\mathcal{O}}$ or simply annealed during training. Annealing the variance to zero allows recovering a Dirac.

Optimizing the Wasserstein terms. To optimize the Wasserstein term of the reward and transition regularizes (Equation 4.6), we follow the same learning procedure as in WAE-MDPs (Chapter 3, Section 3.3): we introduce neural networks $\Gamma_{\spadesuit}$ (for $\spadesuit \in \{\bar{\mathbf{p}}, \Omega\}$) that are trained to attain the supremum of the dual formulation of the Wasserstein distance. To do so, we need enforce the Lipschitzness of $\Gamma_{\spadesuit}$ and, as in WAE-MDPs, we do so via the gradient penalty approach of Gulrajani et al. [2017], leveraging that any differentiable function is 1-Lipschitz if and only if it has gradients with norm at most 1 everywhere.

Finally, we do not directly optimize the total variation distance of $L_{\mathcal{O}}$, but rather the Wasserstein; we take the usual Euclidean distance as metric over Ω that we proved to be Lipschitz equivalent to a distance converging to the discrete metric as the temperature of the latent space goes to zero, assuming Ω is compact (Lemma 20). This allows to eventually recover d_{TV} in the zero-temperature limit.

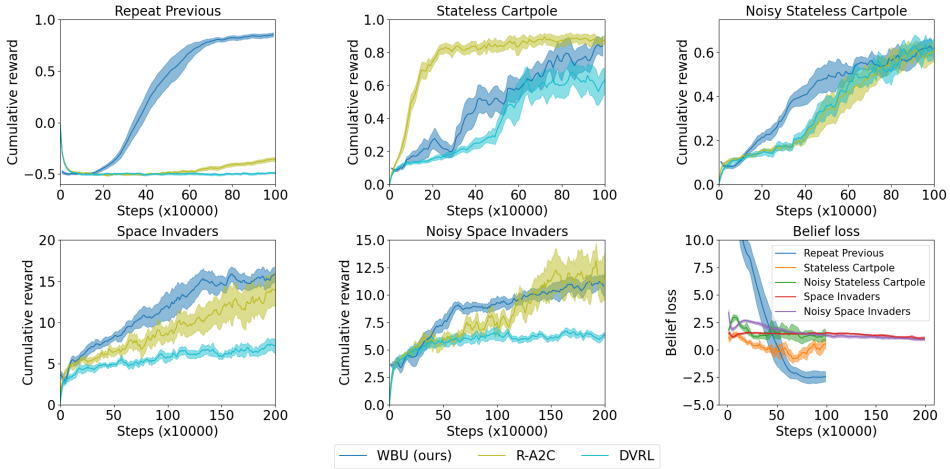


Figure 4.8: Evolution of the (i) undiscounted cumulative return for WBU, R-A2C and DVRL [Igl et al., 2018], and (ii) estimated belief loss during learning for WBU (mean and standard error). We report 5 instances of each algorithm. Appendix C.2 details the hyperparameter search performed.

Learning the policy is enabled by inputting the sub-belief into the policy, while the optimization of the belief encoder parameters by the RL agent is not allowed. Our method is applicable to *any* on-policy algorithm, and we employ A2C (Algorithm 4) in our experiments.

We summarize the learning procedure in Algorithm 7.

4 Experiments

To evaluate our approach, we identify three types of POMDPs: those requiring *long-term memory*, those where *features of the state space are hidden* (and may be inferred from short-term memory), and those with *noisy* observations. Notably, we stress that long-term memory is crucial in POMDPs, whereas short-term memory could be mitigated by stacking frames (e.g., Mnih et al. 2015). We compare our agent to R-A2C and DVRL [Igl et al., 2018] in Figure 4.8, trained in environments from POPGYM [Morad et al., 2023] and our own partially observable version of MINATAR [Young and Tian, 2019].

Memorization. The REPEATPREVIOUS environment involves shuffling two decks of cards at the start of each episode and presenting the agent with a card at each time

Algorithm 7: WASSERSTEIN BELIEF UPDATER

Input: Batch sizes $B_{\text{WBU}}, B_{\text{WAE}}, B_{\text{NEXT}}$; global learning steps N ; no. of model updates per iteration N_{MODEL} ; your favorite collect strategy π_{init} ; replay buffer (RB) $\mathcal{D}$; Lipschitz networks $\Gamma_{\bar{\mathbf{p}}} : \mathcal{S} \rightarrow \mathbb{R}, \Gamma_{\Omega} : \Omega \rightarrow \mathbb{R}$; observation variance network $\bar{\mathcal{O}}_{\sigma}$; and loss weights $w_{\bar{\mathbf{R}}}, w_{\bar{\mathbf{P}}}$

function $\bar{\mathcal{O}}(\cdot \mid \bar{s})$ $\triangleright$ (smooth) Observation Filter

$\mu \leftarrow \bar{\mathcal{O}}_{\mu}(\bar{s}); \sigma \leftarrow \bar{\mathcal{O}}_{\sigma}(\bar{s})$ $\triangleright$ Decode $\bar{s}$; get the standard deviation of the reconstruction

return $\mathcal{N}(\cdot; \mu, \sigma^2)$

collect $\tau = \{s_i, o_i, a_i, r_i, s'_i, o'_i\}_{i=1}^{N_{\text{init}}}$ by executing π_{init} for N_{init} steps; **store** τ in $\mathcal{D}$ $\triangleright$ Use the exploration policy π_{init} to collect transitions and initialize the RB

repeat N times

repeat N_{MODEL} times

$\triangleright$ Update the WAE-MDP model for N_{MODEL} consecutive training steps

for $i \leftarrow 1$ to B_{WAE} **do**

$\langle s_i, o_i, a_i, r_i, s'_i, o'_i \rangle \sim \mathcal{D}$ $\triangleright$ Sample a transition from the RB

$\bar{s}' \leftarrow \phi(\langle s'_i, o'_i \rangle)$ $\triangleright$ Embed $\langle s'_i, o'_i \rangle$ to the latent space

$\bar{o}'_i \sim \bar{\mathcal{O}}(\cdot \mid \bar{s}')$ $\triangleright$ Observe the resulting latent state via $\bar{\mathcal{O}}$

$\mathcal{L}_{\text{WAE}} \leftarrow$ **compute the WAE-MDP loss** on the batch $\{s_i, o_i, a_i, r_i, s'_i, o'_i\}_{i=1}^{B_{\text{WAE}}}$ (Algorithm 6)

Update the WAE-MDP components by minimizing $\mathcal{L}_{\text{WAE}}$

$\mathcal{L}_{\mathcal{O}} \leftarrow 1/B_{\text{WAE}} \cdot \sum_{i=1}^{B_{\text{WAE}}} [\Gamma_{\Omega}(o'_i) - \Gamma_{\Omega}(\bar{o}'_i)]$ $\triangleright$ Observation loss

Update $\Gamma_{\mathcal{O}}$ by maximizing $\mathcal{L}_{\mathcal{O}}$ and enforcing its 1-Lipschitzness w.r.t. metric $d_{\mathcal{O}}$

Update $\bar{\mathcal{O}}_{\sigma}$ by minimizing $\mathcal{L}_{\mathcal{O}}$

for $i \leftarrow 1$ to B_{WBU} **do**

$s_0 \leftarrow s_i; \bar{s}_0 \leftarrow \bar{s}_i; \bar{b}_0 \leftarrow \delta_{s_0}; \beta_0 \leftarrow \beta_i$ $\triangleright \beta_i$ is arbitrary, e.g., zeroes

for $t \leftarrow 0$ to T **do**

$a_t \sim \bar{\pi}(\cdot \mid \beta_t)$ $\triangleright$ Produce the action a_t according to the sub-belief β_t

execute a in $\mathcal{P}$, **receive** reward r_t , and **perceive** the next state-observation $\langle s_{t+1}, o_{t+1} \rangle$

store the transition $\langle s_t, o_t, a_t, r_t, s_{t+1}, o_{t+1} \rangle$ into $\mathcal{D}$

$\beta_{t+1} \leftarrow \Phi^{\text{sub}}(\text{sg}(\beta_t), a_t, o_{t+1})$ $\triangleright$ Update the sub-belief; **sg** refers to “stop gradients”

$\bar{b}_{t+1} \leftarrow \mathcal{M}(\beta_{t+1})$ $\triangleright$ Retrieve the next belief distribution b_{t+1} via the MAF $\mathcal{M}$

$\bar{s}_{t+1} \sim \bar{b}_{t+1}$ $\triangleright$ Believe into the next latent state

$\triangleright$ Marginalize the next latent state distribution w.r.t. the current belief

for $j \leftarrow 1$ to B_{NEXT} **do**

$\bar{s} \sim \bar{b}_t; \mathcal{L}_{\log \bar{\mathbf{P}}}^j \leftarrow [\log \bar{\mathbf{P}}(\bar{s}_{t+1} \mid \bar{s}, a_t) - \log B_{\text{NEXT}}]$

$\mathcal{L}_{\text{KL}}^{i,t} \leftarrow \log \bar{b}_{t+1}(\bar{s}_{t+1}) - \text{LSE}_{j=1}^{B_{\text{NEXT}}} \left(\left\{ \mathcal{L}_{\log \bar{\mathbf{P}}}^j \right\} \right) - \log \bar{\mathcal{O}}(o_{t+1} \mid \bar{s}_{t+1})$

$\triangleright$ Pointwise decomposition of Equation 4.11; LSE refers to the log-sum-exp operation

$\mathcal{L}_{\bar{\mathbf{R}}}^{i,t} \leftarrow |\bar{\mathbf{R}}(\phi(s_t, o_t), a_t) - \bar{\mathbf{R}}(\bar{s}_t, a_t)|$ $\triangleright$ Latent reward regularizer

$\bar{s}' \sim \bar{\mathbf{P}}(\cdot \mid \bar{s}_t, a_t)$ $\triangleright$ Transition to the next latent state from the current believed latent state

$\mathcal{L}_{\bar{\mathbf{P}}}^{i,t} \leftarrow [\Gamma_{\bar{\mathbf{P}}}(\phi(s_{t+1}, o_{t+1})) - \Gamma_{\bar{\mathbf{P}}}(\bar{s}')]^2$ $\triangleright$ Latent transition regularizer

Update $\Gamma_{\bar{\mathbf{P}}}$ by maximizing $\sum_{i=1}^{B_{\text{WBU}}} \sum_{t=0}^{T-1} \mathcal{L}_{\bar{\mathbf{P}}}^{i,t}$ and enforcing its 1-Lipschitzness w.r.t. latent metric $\bar{d}$

Update Φ^{sub} and $\mathcal{M}$ by minimizing $1/(B_{\text{WBU}}+T) \sum_{i=1}^{B_{\text{WBU}}} \sum_{t=0}^{T-1} (\mathcal{L}_{\text{KL}}^{i,t} + w_{\bar{\mathbf{R}}} \cdot \mathcal{L}_{\bar{\mathbf{R}}}^{i,t} + w_{\bar{\mathbf{P}}} \cdot \mathcal{L}_{\bar{\mathbf{P}}}^{i,t})$

Update $\bar{\pi}$ by minimizing the A2C loss on the batch $\{\beta_{0:T}^i, a_{0:T-1}^i, r_{0:T-1}^i\}_{i=1}^{B_{\text{WBU}}}$ $\triangleright$ Algorithm 4

step. The goal is to identify the suit of the card seen 8 time steps earlier. *Our algorithm stands out as the sole method demonstrating mid- to long-term memorization capabilities.* Unlike other methods, notably DVRL which also attempts to learn a belief distribution, WBU provably acquires a suitable representation of the history by learning to maintain a sufficient statistic, thereby explaining its ability to retain past information.

Hidden features. We employ a cart pole scenario (STATELESSCARTPOLE) where velocity components of the system are hidden. R-A2C excels rapidly here, capitalizing on short-term memory to infer velocities from the preceding observation, while DVRL is overtaken. Still, WBU eventually reaches R-A2C final performance. We also explore the SPACEINVADERS environment, where the agent takes command of a cannon with the objective of shooting at groups of moving aliens. In the observation, we intentionally concealed the direction of alien movement and confounded friendly and enemy fires. In this more challenging setting, WBU excels by earning the highest rewards.

Noise. We explore two types of noise. First, we introduce *Gaussian noise* to the observations of STATELESSCARTPOLE. Second, for SPACEINVADERS, *binary noise* is injected via a radar-like mask obscuring the position of each alien with high probability. Hence, the agent must infer their positions based on previous observations. By leveraging its ability to maintain a belief over the noiseless (latent) state space, WBU demonstrates its resilience to noise and swiftly provides superior solutions, whereas R-A2C eventually achieves comparable performance but with more variance.

Belief representation. Ideally, close policy inputs should lead to close values, which would ease its optimization. Theorem 31 provides such a representation guarantee and ensures that the representation induced by Φ captures the value function. To support this, we performed a t-SNE [van der Maaten and Hinton, 2008] on our belief representation at the late stage of training, which projects latent beliefs on a 2D space (Figure 4.9). Interestingly, *latent beliefs clustered together have indeed close values*, in line with Theorem 31. We reported the belief loss throughout the training phases (Figure 4.8). Importantly, unlike other baselines, *our approach*

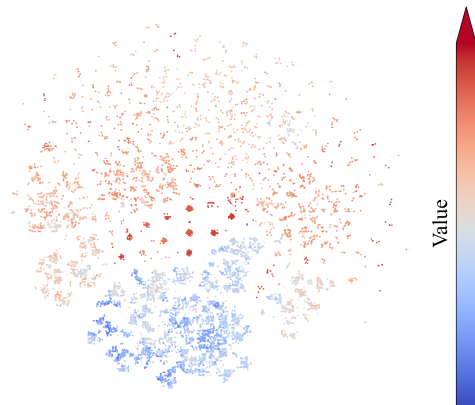


Figure 4.9: Two-dimensional t-SNE of our belief representation for SPACEINVADERS

distinctly separates the optimization of Φ and $\bar{\pi}$. Consequently, the policy optimization does not influence the representation which is solely learned via Φ . The decrease in this loss thus relates to improved representation quality for RL.

5 Concluding Discussion

WBU provides a principled approach that approximates directly the belief update for POMDPs, in contrast to state-of-the-art methods that use the RL objective and regularization to attempt to turn the history into a sufficient statistic. By learning the belief and its update rule, we provide strong guarantees on the quality of the belief, its ability to condition the optimal value function, and ultimately, the effectiveness of our algorithm. Our theoretical analysis and experimental results demonstrate the potential of our approach. Overall, WBU provides a promising new direction for RL in POMDPs, with potential applications in a wide range of settings where decision-making is complicated by uncertainty and partial observability, or when guarantees on the agent behaviors are required.

The theory we developed is not limited to the algorithm proposed in this chapter. It opens diverse avenues for future work, e.g., on formally verifiable policies for POMDPs, by leveraging the guarantees presented in our framework. We also leave the study and the adaptation of our framework under relaxed assumptions (e.g., in settings akin to the work of Lambrechts et al., 2023) to future work. In addition, Theorem 30 enables policy optimization through planning in the learned model, as demonstrated in successful model-based RL methods (e.g., Hafner et al. 2021). Scaling to high-dimensional observations (e.g., images) may potentially be computation-intensive due to observation filtering. For this further challenge, we suggest either to modify the WAE-MDP framework by using a stochastic decoder (see Tolstikhin et al., 2018, e.g., via PixelCNN, van den Oord et al., 2016), or learning a lower-dimensional latent observation space synced with the policy with a normalized or (relaxed) discrete prior (e.g., via a WAE-GAN, Tolstikhin et al., 2018).

Synthesis from Deep Reinforcement Learning Components

In Chapter 3, we mainly focused on model checking deep reinforcement learning policies. In this chapter, we rather focus on synthesis: we start from a specification and the goal is to construct a controller which satisfies the specification by design. For a detailed discussion on synthesis, we refer the reader to the introduction of Section 3 in Chapter 2. Notably, we consider the case where the environment is too large to be modeled explicitly, in which case standard synthesis algorithms cannot be applied. To address this challenge, we propose to combine deep RL with reactive synthesis to construct the controller. Deep RL and synthesis are two complementary approaches to obtaining control policies with different strengths and weaknesses. We recall the main differences between the two approaches — benefits and drawbacks — and motivate the introduction of a unified framework, alleviating the limitations of each technique while preserving their advantages.

On the one hand, one may consider RL to learn the control policy. In RL, the designer chooses how rewards are issued, and the RL agent is trained to learn a control policy that optimizes the cumulative sum of rewards. In particular, deep RL shows success in domains of increasing size, often surpassing human capabilities. On the downside, designing a reward function is a challenging engineering task in which the designer needs to both train the agent to exhibit desired behavior and train it efficiently. While the satisfaction of specifications may be encoded as binary sparse signals issued along the interaction, sparse rewards are challenging for RL agents. For instance, consider a

robot obtaining a reward for exiting a building. Since exiting by random exploration is improbable, training is time-consuming and might even fail in simple environments. In practice, dealing with sparse rewards can be alleviated by guiding the agent to the intended behavior through intricate techniques [Liu et al., 2022a]. This in turn, adds more problems as the “desired behavior” is specified via rewards, and reward engineering can lead to behavior that may not align with the user’s intentions.

On the other hand, synthesis constructs an optimal policy based on a model of the environment and an objective specified as a logical formula. In contrast to DRL, synthesis provides guarantees that the policy satisfies the specification and allows users intuitive and natural specification languages. For example, exiting the building in linear temporal logic is $\Diamond\text{exit}$. The reliance on an explicit model of the environment is its key disadvantage; the technique struggles with scalability and domains in which dynamics are partially known.

In this chapter, we *propose a framework that aims to gain the best of both worlds*. We introduce a novel approach to obtaining policies in large environments when only limited information about their high-level, abstract structure is available. We require minimal prior knowledge on this structure: the input is a *map* given as a *graph*, where each vertex is populated by an (unknown) *room*, modeled as an MDP. We argue that this is a natural requirement in many domains. Suppose the building in which the robot operates is a warehouse, where it needs to deliver a package. The warehouse is divided into rooms amid moving obstacles (e.g., forklifts, workers, or other robots). While it is infeasible to provide a model describing all the possible interactions the agent may have within the warehouse and the dynamics of the moving obstacles, one can reasonably assume a *map* is provided.

In each room, we train DRL policies to complete short-horizon room-specific tasks, e.g., safely exit a room. Based on the DRL policies, we synthesize a high-level *planner* to solve a long-horizon task given on the map, e.g., deliver the package at a target location. We call the combination of low-level policies and the high-level planner a *hierarchical controller*.

We list advantages of our framework. First and foremost, it *provides guarantees on the operation of the controller*. As mentioned, it enjoys the best of both worlds: it *enables synthesis with theoretical guarantees in large partially-known environments*. It allows a “separation of concerns”: reward engineering is only done locally while *high-level tasks are given in an intuitive specification language*. In addition, it offers a remedy for the notorious challenges of sparse rewards in RL. Interestingly, it enables *reusability*: the policies in rooms and their guarantees are reusable across similar rooms and when the high-level task or structure change.

We stress that, while our approach naturally fits the given example, it is not limited to such navigation scenarios. Others include probabilistic programs (e.g., network protocols and job scheduling, Junges and Spaan, 2022), systems that can be formalized as string diagrams (e.g., dice games, Watanabe et al., 2024), and more general software systems

relying on libraries of reusable components such as drivers in an operating system [Ryzhyk et al., 2009]. Notably, our approach can be seen as *post-hoc* to *hierarchical RL* [Barto and Mahadevan, 2003], where the high-level structure is eventually learned and fixed, but not yet the low-level components.

We complement our theoretical results with a case study which illustrates the feasibility of the approach. We consider a domain of parameterizable size in which an agent needs to reach a distant location while avoiding moving adversarial obstacles with stochastic dynamics. We show that DQN struggles to find a policy in our domain, even with reward shaping. In the rooms, we demonstrate our novel procedure for training concise latent policies directly. We synthesize a planner based on the latent policies and show the following results. First, our high-level controller achieves high success probability, demonstrating that our approach overcomes the challenge of sparse rewards. Second, the values predicted in the latent model are close to those observed, demonstrating the quality of our automatically constructed model. Third, we complement the latter with probably approximately correct (PAC) bounds on the abstraction quality.

Related work. We compare with other approaches to obtain high-level controllers. In hierarchical RL or the *options* framework [Sutton et al., 1999b], the high-level component learns a policy over subgoals and a low-level component learns to achieve them. Both are learned concurrently, while for us the map is given — our problem is post-hoc to learning the high-level component. Furthermore, the reusability of our low-level components and the fact that the choice of the low-level policy cannot be resolved in a Markovian fashion (i.e., without memory, details in Section 3) is a distinction from option-inspired approaches. Another recent approach is the CLAPS algorithm [Žikelić et al., 2023] to learn low-level components with correctness witnesses. However, CLAPS focuses on stochastic feedback loops, where both the transition function and the controller are assumed Lipschitz continuous in the state space and respectively restricted to re-parameterizable distributions and deterministic, stationary policies. In contrast, we consider MDPs with (intractable) finite spaces and general policies. Moreover, the low-level components of CLAPS are not related to the high-level structure of the environment but rather to the considered logical specification.

The key challenge in planner synthesis is to obtain a ranking criterion, i.e., an estimate on the policy’s success probability. DRL outputs a neural network (NN), which is too large to incorporate in a synthesis procedure [Kazak et al., 2019], and we assume no knowledge of transition probabilities in rooms (rather, only simulator access). Our approach draws from the common framework of training an NN, distilling a concise *latent* model, and applying formal reasoning to the latent model (see, e.g., Chapter 3 and Ernst et al., 2005; Alamdari et al., 2020; Bastani et al., 2018; Tornblom and Nadjm-Tehrani, 2019). Note that the latter only gives guarantees if distillation provides error bounds on the abstraction induced by the latent model. In contrast, our method is of independent interest and trains

a latent policy directly, thereby circumventing the need for model distillation; it outputs a latent MDP together with a mapping from concrete environment states to abstracted states, and PAC guarantees on the latent policy value. We stress that the abstraction is learned, unlike in the works of Roderick et al. [2018]; Jothimurugan et al. [2021b].

The work developed in this chapter is concerned with reach-avoid objectives in RL. Such objectives can be specified in LTL or VLC (cf. Chapter 1, Sections 3.1 and 3.5). While hierarchical RL is a notoriously difficult problem [Kulkarni et al., 2016], LTL objectives add intractability (cf. Chapter 1, Section 3.4) and only allow for PAC guarantees if the MDP structure is known [Fu and Topcu, 2014]. Besides the mentioned work of Žikelić et al. [2023], high-level controllers are used in recent works such as combining a planner with a low-level learned policy and a safety shield [Xiong et al., 2022]; however, the ad-hoc integration of the learned component does not provide guarantees. Moreover, Nayak et al. [2023] obtain low-level controllers via reactive synthesis, which does not scale to complex scenarios. Regarding safety objectives, RL remains intractable [Alur et al., 2022]. For guaranteed safety, one can synthesize a shield that blocks unsafe actions [Alshiekh et al., 2018; Könighofer et al., 2022]. In our setting, we would need shields for the low-level policies; constructing them would require full access to the rooms’ models (which are too large). Approaches encouraging but not ensuring safety use constrained policy optimization [Achiam et al., 2017], safe padding in small steps Hasanbeig et al. [2020], time-bounded safety [Giacobbe et al., 2021], safety-augmented MDPs [Sootla et al., 2022], differentiable probabilistic logic [Yang et al., 2023b], or distribution sampling [Badings et al., 2023].

Origin of the results. This chapter presents original and unpublished results, which can be found in the preprint [Delgrange et al., 2024].

1 Problem Setting

MDPs and initial distribution. In this chapter, we consider environments modeled as (very large) MDPs with finite state and action spaces. Let $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathbf{P}, \mathcal{R}, \mathbf{AP}, \ell, \mathbf{I}, \gamma \rangle$ be an MDP. For the sake of simplicity, notice that we replaced the initial state of the MDP with an initial distribution over states, $\mathbf{I} \in \Delta(\mathcal{S})$. Note that this formulation is equivalent to our standard definition of MDPs (Definition 5): to pass from an MDP with initial distribution $\mathbf{I}$ to an MDP with an initial state, it simply suffices to add a new initial dummy state s_I to the state space and define its transition function as $\mathbf{P}'(\cdot \mid s_I, a) = \mathbf{I}$ for all $a \in \mathcal{A}$, then make all rewards shift by γ to take this additional transition into account, i.e., $\mathcal{R}' = \mathcal{R}/\gamma$.

When $\mathbf{AP}$ and ℓ are omitted from the definition, we may simply consider the *natural labeling* of $\mathcal{M}$ (cf. Definition 5), i.e., $\mathbf{AP} = \mathcal{S}$ and $\ell(s) = \{s\}$ for all $s \in \mathcal{S}$.

Reach-avoid objectives. We specifically focus on reach-avoid objectives, which are a combination of reachability and safety specifications: the agent must reach a target set $T \subseteq AP$ while avoiding a set of unsafe labels $B \subseteq AP$. Precisely, we consider the VLC formula

$$\varphi(T, B) = \lambda x. ((\text{avoid} \wedge \gamma \cdot \circ x) \vee (\text{reach} \wedge \text{avoid})) \quad (5.1)$$

where $\text{avoid} = \bigwedge_{b \in B} \neg b$ and $\text{reach} = \bigvee_{t \in T} t$. Notably, as the discount factor goes to one, the reach-avoid objective translates to the LTL formula $\text{avoid} \cup (\text{reach} \wedge \text{avoid})$: the value of every state $s \in S$ for a given policy π w.r.t. the objective $\varphi(T, B)$, written $V^\pi(s, \varphi(T, B))$, reduces to the probability of satisfying the objective starting from s and following π as $\gamma \rightarrow 1$, i.e., $\lim_{\gamma \rightarrow 1} V^\pi(s, \varphi(T, B)) = \mathbb{P}_\pi(\tau \models \text{avoid} \cup (\text{reach} \wedge \text{avoid}))$. When clear from the context, we may simply write φ for $\varphi(T, B)$. We are particularly interested in the values obtained from the beginning of the execution, which we denote by $V_1^\pi(\varphi) = \mathbb{E}_{s_0 \sim I} [V^\pi(s_0, \varphi)]$.

1.1 Problem Formulation

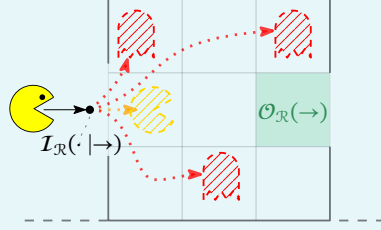
In this section, we formally model a hierarchical environment and state the problem of hierarchical controller synthesis.

Intuitively, we consider an MDP given by a high-level *map* as an undirected graph, each of whose every vertex is associated with a “low-level” MDP called a *room*. A hierarchical controller operates as follows. In each room, we assume access to a collection of *low-level policies*, each optimizing a local (room) reach-avoid objective. We obtain these low-level policies using DRL since the rooms may be large and of unknown structure. When transitioning to a new room, a high-level *planner* selects the next low-level policy. We synthesize this planner to optimize a high-level reachability objective. The challenge with this hierarchical structure is to obtain concise low-level policies with performance guarantees to enable a feasible synthesis procedure.

Hierarchical model. A room $\mathcal{R} = \langle S_{\mathcal{R}}, \mathcal{A}_{\mathcal{R}}, P_{\mathcal{R}}, \text{Dir}_{\mathcal{R}}, I_{\mathcal{R}}, \mathcal{O}_{\mathcal{R}}, AP, \ell_{\mathcal{R}} \rangle$ consists of states, actions, transition, and labeling functions $S_{\mathcal{R}}, \mathcal{A}_{\mathcal{R}}, P_{\mathcal{R}}$, and $\ell_{\mathcal{R}}$ as in an MDP, a set of *directions* $\text{Dir}_{\mathcal{R}}$, an *entrance function* $I_{\mathcal{R}} : \text{Dir}_{\mathcal{R}} \rightarrow \Delta(S_{\mathcal{R}})$ taking a direction from which the room is entered and producing an *initial distribution over states* of the room, and an *exit function* $\mathcal{O}_{\mathcal{R}} : \text{Dir}_{\mathcal{R}} \rightarrow 2^{S_{\mathcal{R}}}$ returning a set of *exit states* from the room in a given direction $d \in \text{Dir}_{\mathcal{R}}$. We assume states are assigned to at most one exit, i.e., if $s \in \mathcal{O}_{\mathcal{R}}(d)$ and $s \in \mathcal{O}_{\mathcal{R}}(d')$, then $d' = d$. We further see each direction as an atomic proposition, with $\text{Dir}_{\mathcal{R}} \subseteq AP$: this allows to label exit states with d so that $s \in \mathcal{O}_{\mathcal{R}}(d)$ if and only if $d \in \ell_{\mathcal{R}}(s)$.

Example 19: Room

Consider the grid world of the right figure as a room $\mathcal{R}$ populated by an adversary  whose position is encoded in $S_{\mathcal{R}}$ and behavior in $P_{\mathcal{R}}$. The position of  depends on the direction from which the agent enters $\mathcal{R}$. The agent  enters from the left in direction $\rightarrow$ to the states of $\mathcal{R}$ distributed according to the entrance function $I_{\mathcal{R}}(\cdot \mid d \Rightarrow \rightarrow)$ (the tiling patterns highlight its support). Precisely, while  is deterministically sent to the leftmost cell (yellow tiling), $I_{\mathcal{R}}$ allows to (probabilistically) model the possible positions of  when entering the room (red tiling). When reaching the green area, depicting states from $\mathcal{O}_{\mathcal{R}}(\rightarrow)$,  may exit $\mathcal{R}$ by the right direction $\rightarrow$.



For a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ with vertices $\mathcal{V}$ and *undirected edges* $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Since the graph is undirected, we have $\langle v, u \rangle \in \mathcal{E}$ if and only if $\langle u, v \rangle \in \mathcal{E}$. The *neighbors* of $v \in \mathcal{V}$ are $Neighbors(v) = \{u \in \mathcal{V} \mid \langle u, v \rangle \in \mathcal{E}\}$ and the outgoing edges from v are $out(v) = \{\langle u, t \rangle \in \mathcal{E} : u = v \text{ or } t = v\}$. A *path* in $\mathcal{G}$ corresponds to a finite sequence of vertices $v_0, v_1, \dots, v_n \in \mathcal{V}^*$ such that $\langle v_i, v_{i+1} \rangle \in \mathcal{E}$ for all $i < n$.

A *hierarchical model* $\mathcal{H} = \langle \mathcal{G}, Rooms, \ell, v_0, \langle d_0, d_1 \rangle, \gamma \rangle$ consists of a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, a collection of rooms $Rooms$, a labeling $\ell : \mathcal{V} \rightarrow Rooms$ of each vertex $v \in \mathcal{V}$ with a room $\ell(v)$ such that the directions are $Dir_{\ell(v)} = out(v)$, an initial room $v_0 \in \mathcal{V}$, directions $d_0, d_1 \in out(v_0)$ from and in which v_0 is respectively entered and must be left, and a discount factor $\gamma \in (0, 1)$. Note that we allow multiple copies of a room, i.e., there can be $u \neq v$ with $\ell(u) = \ell(v)$.

For the remainder of the section, fix a hierarchical model $\mathcal{H} = \langle \mathcal{G}, Rooms, \ell, v_0, \langle d_0, d_1 \rangle, \gamma \rangle$. The explicit MDP $\mathcal{M}$ corresponding to $\mathcal{H}$ is obtained by, intuitively, “stitching” MDPs $\mathcal{R} \in Rooms$ corresponding to neighboring rooms. Figure 5.1 illustrates the different layers of a hierarchical model and the corresponding explicit MDP. Formally, we define the explicit MDP as the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathbf{P}, \mathbf{R}, \mathbf{I}, \gamma \rangle$, where $\mathcal{S} = \{\langle s, v \rangle : s \in S_{\ell(v)}, v \in \mathcal{V}\}$, $\mathcal{A} = \bigcup_{\mathcal{R} \in Rooms} \mathcal{A}_{\mathcal{R}} \cup \{a_{exit}\}$, and $\mathbf{P}$ and $\mathbf{I}$ are defined as follows. The initial distribution $\mathbf{I}$ simulates entering $\mathcal{M}$ in the room $\ell(v_0)$ from direction d_0 . Thus, for every $s \in S_{\ell(v_0)}$, we have $\mathbf{I}(s, v_0) = I_{\ell(v_0)}(s \mid d_0)$. The transitions $\mathbf{P}$ coincide with $\mathbf{P}_{\mathcal{R}}$ for non-exit states. Let $d = \langle v, u \rangle \in \mathcal{E}$ so that $v \in Neighbors(u)$; $\mathcal{O}_{\mathcal{R}}(d)$ are the exit states in room $\mathcal{R}$ associated with v in direction d , and $I_{\ell(u)}(d)$ is the distribution over initial states in the room associated with u in direction d . The successor state of $s \in \mathcal{O}_{\mathcal{R}}(d)$ follows the distribution $I_{\ell(u)}(d)$ when the action a_{exit} is

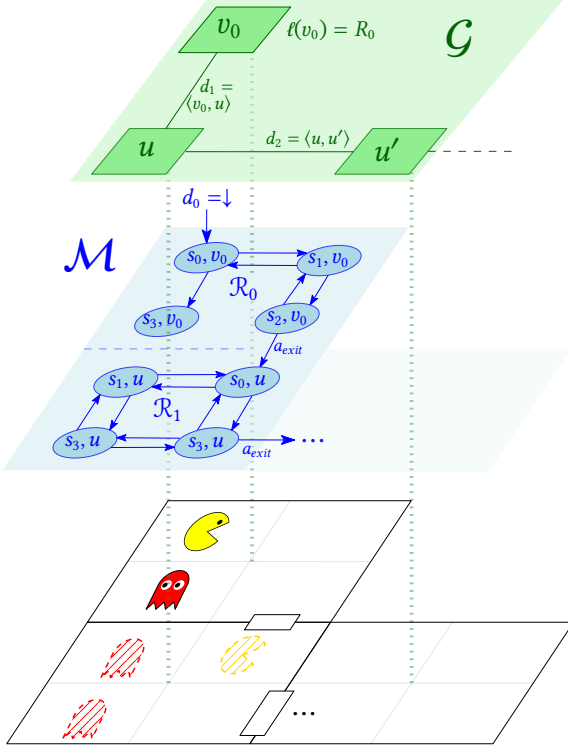


Figure 5.1: Example showing the different layers of a hierarchical model. Bottom: a snapshot of a grid world environment with several rooms (two depicted here) with one controllable agent  and moving adversaries . Middle: an explicit MDP $\mathcal{M}$ composed of two rooms $\mathcal{R}_0$ and $\mathcal{R}_1$. Arrows between states depict (some) possible transitions. Top: a graph $\mathcal{G}$ composed of vertices $\mathcal{V} = \{v_0, u, u', \dots\}$ and edges $E = \{d_1, d_2, \dots\}$ for transitions between rooms. The agent  enters $\langle s_0, v_0 \rangle \in \mathcal{S}$ ($\ell(v_0) = \mathcal{R}_0$ and $s_0 \in \mathcal{S}_{\mathcal{R}_0}$) from the top ($d_0 = \downarrow$) and must avoid  and exit $\mathcal{R}_0$ through the door on the bottom right: $\mathcal{O}_{\mathcal{R}_0}(d_1 = \langle v_0, u \rangle) = \{s_2\}$. The function $\mathcal{I}_{\mathcal{R}_1}(\cdot \mid d_1 = \langle v_0, u \rangle)$ produces an initial probability distribution over the possible positions of  and  when entering $\mathcal{R}_1$ (tiling patterns), in the same spirit as in Example 19.

chosen. Note that each trajectory τ in $\mathcal{M}$ corresponds to a unique path $path(\tau)$ in $\mathcal{G}$ that traverses the sequence of rooms τ visits.

High-level reach and low-level reach-avoid objectives. The *high-level reachability objective* that we consider is $\Diamond G$, where $G \subseteq \mathcal{V}$ is a collection of *goal vertices* in the high-level graph of $\mathcal{H}$. A path τ in $\mathcal{M}$ satisfies $\Diamond G$ if and only if $path(\tau)$ visits a vertex v in G , i.e., τ visits the room located in v .

We define the *low-level safety objective* for the discount factor γ over the states of the rooms in $Rooms$. For each room $\mathcal{R}$, let $\mathbf{B}_{\mathcal{R}} \subseteq \mathbf{AP}$ be a set of unsafe (“bad”) labels that

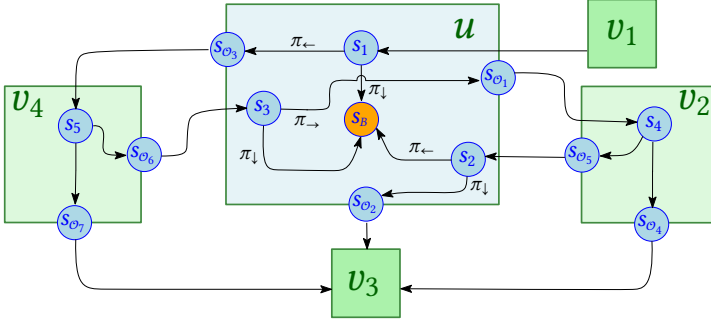


Figure 5.2: Hierarchical model $\mathcal{H}$ where a set collection of low-level policies Π is fixed; all probability distributions are deterministic. Graph vertices are depicted by squares and room states by circles. States of the form $s_{\mathcal{O}}$ (blue circles) are exit states. Subscripts $\rightarrow, \downarrow, \leftarrow$ respectively indicate directions $\langle u, v_2 \rangle, \langle u, v_3 \rangle$, and $\langle u, v_4 \rangle$ in the room $\ell(u)$. $\mathbf{B} = \ell(s_{\mathbf{B}})$ (orange circle) and $G = \{v_3\}$ are the set to avoid and the high-level goal, respectively. The set of low level policies include those from room $\ell(u)$, i.e., $\{\pi_{\leftarrow}, \pi_{\downarrow}, \pi_{\rightarrow}\} \subseteq \Pi$. A hierarchical controller $\pi = \langle \omega, \Pi \rangle$ selects which low-level policy from Π to execute in each room, according to the directions chosen by the planner ω .

the agent must avoid. Moreover, given a direction $d \in \text{Dir}_{\mathcal{R}}$, we define the reach-avoid objective $\varphi_{\mathcal{R}}^d(\mathbf{T}_{\mathcal{R}} = \{d\}, \mathbf{B}_{\mathcal{R}})$, namely exiting $\mathcal{R}$ in direction d while avoiding bad states.¹

Hierarchical control. We define a *high-level planner* $\omega : \mathcal{V}^* \rightarrow \mathcal{E}$ and a collection of low-level policies Π such that, for each room $\mathcal{R} \in \text{Rooms}$ and a direction $d \in \text{Dir}_{\mathcal{R}}$, Π contains a policy $\pi_{\mathcal{R},d}$ for the objective $\varphi_{\mathcal{R}}^d$. The pair $\pi = \langle \omega, \Pi \rangle$ is a *hierarchical controller* for $\mathcal{H}$, defined inductively as follows. Consider an initial vertex $v_0 \in \mathcal{V}$. Let $d_0 = \omega(\epsilon), d_1 = \omega(d_0) \in \text{out}(v_0)$. Initially, the room $\ell(v_0)$ is entered according to the distribution $\mathcal{I}_{\ell(v_0)}(\cdot | d_0)$. Then, the control proceeds according to the policy $\pi_{\ell(v_0),d_1}$. Let τ be a finite trajectory in $\mathcal{H}$ ending in $s \in \mathcal{S}_{\mathcal{R}}$, for some room $\mathcal{R} = \ell(v)$. If s is not an exit state of $\mathcal{R}$, then control proceeds according to a policy $\pi_{\mathcal{R},d}$ with $d = \langle v, u \rangle$ and $u \in \text{Neighbors}(v)$. Suppose that s is an exit state in direction d and that path(τ) ends in v , i.e., $s \in \mathcal{O}_{\mathcal{R}}(d)$. Then a_{exit} is taken in s and the next state is an initial state in $\mathcal{R}' = \ell(u)$ drawn from $\mathcal{I}_{\mathcal{R}'}(\cdot | d)$. The planner must choose the direction $d' = \omega(\text{path}(\tau) \cdot u) \in \text{out}(u)$ from which to exit $\mathcal{R}'$. Control of $\mathcal{R}'$ finally proceeds according to the low-level policy $\pi_{\mathcal{R}',d'}$.

Note that π can be simply considered as a policy in the explicit MDP $\mathcal{M}$. An example of a hierarchical model with a fixed collection of low-level policies can be found in Figure 5.2.

¹we set $\mathbf{T}_{\mathcal{R}}$ to $\{d\}$ since states from $\mathcal{R}$ are labeled with d if and only if they are exit states from $\mathcal{O}_{\mathcal{R}}(d)$.

Problem statement

Given a hierarchical model $\mathcal{H} = \langle \mathcal{G}, \text{Rooms}, \ell, v_0, \langle d_0, d_1 \rangle, \gamma \rangle$, we define the high-level objective as the tuple $\Omega = \langle \Diamond G, \omega \rangle$ with goal $G \subseteq \mathcal{V}$ and a set of low-level reach-avoid objectives $\omega = \{ \varphi_{\mathcal{R}}^d : \mathcal{R} \in \text{Rooms}, d \in \text{Dir}_{\mathcal{R}} \}$, construct a hierarchical controller $\pi = \langle \omega, \Pi \rangle$ maximizing the value of reaching a goal vertex in $\mathcal{G}$ while avoiding the bad states in each room.

2 Obtaining Latent Low-Level Policies using DRL

There are fundamental challenges in reasoning about policies obtained by DRL, which are typically represented by large NNs. In this section, we develop a novel unified DRL procedure called *WAE-DQN*, which outputs a latent model together with a concise policy accompanied by PAC guarantees. We believe that WAE-DQN is of independent interest.

2.1 Latent Space Models and Guarantees

We consider learning a latent MDP $\overline{\mathcal{M}} = \langle \overline{S}, \overline{A}, \overline{\mathcal{R}}, \text{AP}, \overline{\ell}, \overline{\mathbf{I}} \rangle$ for each room $\mathcal{R}$, associated with an embedding ϕ from concrete to latent states. This way, any latent policy for $\overline{\mathcal{M}}$ can be lifted to the room $\mathcal{R}$: let $s \in S_{\mathcal{R}}$, then it suffices to process s through ϕ to execute $\bar{\pi}$ in $\mathcal{R}$ by generating actions via $a \sim \bar{\pi}(\cdot \mid \phi(s))$ (see Chapter 3, Section 1 and Figure 3.3). As in the previous chapters, we respectively write $V^{\bar{\pi}}$ and $\bar{V}^{\bar{\pi}}$ for the value function of the original environment and the latent model, when they both operate under the same latent policy $\bar{\pi}$.

Then, the goal is to learn the latent policies $\bar{\pi}$ populating the collection of low-level policies Π of the hierarchical controller. This way, each policy from Π is associated with the guarantees we derived in Chapter 3, Sections 1.2 and 1.3 (refer to Table 3.1 for a summary of the guarantees we provide by learning latent models). Notably, the guarantees rely on bounding the difference of values that the agent can achieve in the latent models and the concrete environments (here, the rooms). To achieve this, we have shown in Chapter 3, Section 1.2 that this is made possible by learning to minimize the errors in the transition functions of the original and latent model (called the *transition loss*, cf. Definition 1.1). We quickly recall some results that will be useful in the subsequent sections.

Let φ be a reach-avoid objective. Since the spaces are assumed discrete in this chapter, one can see every probability distribution as a vector of probabilities in the simplex $\Delta^n = \{ \mathbf{x} \in \mathbb{R}^n : \sum_{i=0}^n x_i = 1 \text{ and } x_i \geq 0 \text{ for all } i = 1, \dots, n \}$ where n is the number of states in the environment under consideration. Then, assuming $\mathbf{P}$ is the transition function of the

environment, the transition loss obtained under a latent policy $\bar{\pi}$ is given by

$$L_P = 1/2 \cdot \mathbb{E}_{s,a \sim \xi_{\bar{\pi}}} \|\phi \mathbf{P}(\cdot | s, a) - \bar{\mathbf{P}}(\cdot | \phi(s), a)\|_1. \quad (5.2)$$

Minimizing L_P ensures that the difference of values between the latent and concrete models is also minimized (Theorem 23):

$$|V_1^{\bar{\pi}}(\varphi) - \bar{V}_1^{\bar{\pi}}(\varphi)| \leq \frac{L_P}{\xi_{\bar{\pi}}(s_{reset})(1 - \gamma)},$$

where $1/\xi_{\bar{\pi}}(s_{reset})$ corresponds to the average episode length under $\bar{\pi}$. This result is PAC estimable (Theorem 25). Furthermore, minimizing L_P ensures that original states that are clustered together in the latent space have close value for the reach-avoid objective (Theorem 24): for all states s_1, s_2 , $\phi(s_1) = \phi(s_2)$ implies

$$|V^{\bar{\pi}}(s_1, \varphi) - V^{\bar{\pi}}(s_2, \varphi)| \leq \frac{\gamma L_P}{1 - \gamma} \cdot \left(\frac{1}{\xi_{\bar{\pi}}(s_1)} + \frac{1}{\xi_{\bar{\pi}}(s_2)} \right). \quad (5.3)$$

To enable those guarantees, we need to ensure that one can access a stationary distribution for every room abstracted through a latent space model (Assumption 2). For simplicity, we assume that the low-level policies are learned in the rooms via episodic simulations, which guarantees the existence of a unique stationary distribution per room (Corollary 1).

Assumption 11: Rooms are episodic

Each room $\mathcal{R} \in \text{Rooms}$ is episodic (Definition 16). In particular, since all rooms $\mathcal{R}$ make use of an initial distribution $\mathbf{I}_{\mathcal{R}}$, the transition function need satisfy $\mathbf{P}_{\mathcal{R}}(\cdot | s_{reset}, a) = \mathbf{I}_{\mathcal{R}}$ for all $a \in \mathcal{A}_{\mathcal{R}}$.

Remark 29: Reachability in ergodic rooms

Assumption 11 implies the ergodicity of the rooms and the latent model (Theorem 7). In ergodic MDPs, each state is almost surely visited infinitely often (cf. Chapter 2, Section 1.4). Therefore, for unconstrained reachability goals ($\mathbf{B} = \emptyset$), while discounting still provides insights into how quickly the objective is achieved, optimizing the values associated with reaching the target $\mathbf{T}$ *before the episode concludes* (thus, $\mathbf{B} = \{\text{reset}\}$) is often more appealing.^a This involves finding a policy

π that maximizes $V_I^\pi(\varphi(\mathbf{T}, \mathbf{B} = \{\text{reset}\}))$. In essence, this is exactly how an RL agent is trained: learning to fulfill the low-level objective before the episode concludes.

^areset is a special label for s_{reset} , see Assumption 5.

2.2 Obtaining Latent Policies During Training

Algorithm 8: WAE-DQN

Input: steps T , model updates N , batch sizes $B_{\text{WAE}}, B_{\text{DQN}}$, and $\alpha, \epsilon \in (0, 1)$; Deep Q-Network Q with params. θ_{DQN} , state embedding ϕ with params. ι , and latent model $\bar{\mathcal{M}}$ with params. θ_{WAE} ;

Initialize the target parameters: $\langle \hat{\iota}, \hat{\theta}_{\text{DQN}} \rangle \leftarrow \text{copy the parameters } \langle \iota, \theta_{\text{DQN}} \rangle$

Initialize replay buffer $\mathcal{D}$ with transitions from random exploration of $\mathcal{M}$

for $t \in \{1, \dots, T\}$ **with** $s_0 \sim \mathbf{I}$ **do**

 Embed s_t into the latent space: $\bar{s} \leftarrow \phi(s_t)$

$\triangleright$ Your favorite exploration strategy (here, ϵ -greedy)

 Choose action a_t : $\begin{cases} \text{w.p. } (1-\epsilon), \text{ define } a_t = \arg \max_a Q(\bar{s}, a), \text{ and} \\ \text{w.p. } \epsilon, \text{ draw } a_t \text{ uniformly from } \mathcal{A} \end{cases}$

 Execute a_t in the environment $\mathcal{M}$, receive reward r_t , and observe s_{t+1}

 Store the transition in the replay buffer: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r_t, s_{t+1})\}$

repeat N **times**

 Sample a batch of size B_{WAE} from $\mathcal{D}$: $\mathcal{X} \leftarrow \{(s, a, r, s')_i\}_{i=1}^{B_{\text{WAE}}} \sim \mathcal{D}$

 Update ι and θ_{WAE} on the batch $\mathcal{X}$ by minimizing the WAE-MDP loss

 (including L_P) for the latent policy $\bar{\pi}$ $\triangleright$ cf. Algorithm 6

for $i \in \{1, \dots, B_{\text{DQN}}\}$ **do**

 Sample a transition from $\mathcal{D}$: $s, a, r, s' \sim \mathcal{D}$

 Compute the target: $\hat{y} \leftarrow r + \gamma \max_{a' \in \mathcal{A}} Q(\phi(s'); \hat{\iota}, a'; \hat{\theta}_{\text{DQN}})$

 Compute the DQN loss (Equation 2.22): $L_i \leftarrow (Q(\phi(s; \iota), a; \theta_{\text{DQN}}) - \hat{y})^2$

 Update ι and θ_{DQN} by minimizing $1/B_{\text{DQN}} \sum_{i=1}^{B_{\text{DQN}}} L_i$

 Update the target params.: $\hat{\iota} \leftarrow \alpha \cdot \iota + (1 - \alpha) \cdot \hat{\iota}$; $\hat{\theta} \leftarrow \alpha \cdot \theta_{\text{DQN}} + (1 - \alpha) \cdot \hat{\theta}$

return $\phi, \bar{\mathcal{M}}$, and $\bar{\pi}$

In this section, we introduce WAE-DQN, a DRL procedure that trains a latent MDP over finite state-action spaces and policy. WAE-DQN alternates between the optimization of a latent policy via DQN (Algorithm 3, details in Chapter 2, Section 2.2) and representation

learning, via a WAE-MDP (Algorithm 6, details in Chapter 3, Section 3). The detailed learning procedure is described in Algorithm 8.

Intuitively, a WAE-MDP and a (latent) DQN policy are learned in a round-robin fashion: the WAE-MDP produces the input representation (induced by ϕ) that the DQN agent uses to optimize its policy $\bar{\pi}$. At each step $t = 1, \dots, T$, the environment is explored via some strategy to collect transitions in a replay buffer. Each training step consists of two rounds of optimizations. The first one optimizes the parameters of $\bar{P}$ and ϕ . Second, we optimize the DQN's parameters to learn the policy as in DQN. We further allow DQN to backpropagate gradients through ϕ . We make use of a *target embedding function* $\hat{\phi}$, for stability purposes (in the same spirit as Zhang et al., 2021). This is consistent with DQN's target network approach: the weights of $\hat{\phi}$ are periodically synchronized with those of the original embedding function ϕ . Then, $\hat{\phi}$ is paired with the DQN's target network, which allows to avoid oscillations and dramatic shifts in the representation.

This learning procedure yields two properties (implied by the representation quality guarantees of WAE-MDPs, cf. Equation 5.3): (i) the embedding function ϕ groups states with close values, easing the learning of $\bar{\pi}$; (ii) $\bar{\pi}$ prescribes the same actions for states exhibiting similar behaviors, enhancing robustness to environment perturbations. The direct learning of $\bar{\pi}$ has two key benefits: (i) it removes the need for alternating between policy optimization and distillation phases, simplifying the process; (ii) it enables the use of the same latent space for different rooms with similar structures.

3 Obtaining a High-Level Planner

We fix $\mathcal{H} = \langle \mathcal{G}, \ell, \text{Rooms}, v_0, \langle d_0, d_1 \rangle, \gamma \rangle$ as hierarchical model. In this section, we construct a high-level planner ω for $\mathcal{H}$ based on a collection of low-level latent policies Π (cf. Figure 5.3). Recall that $\varphi_{\mathcal{R}}^d$ denotes exiting room $\mathcal{R} \in \text{Rooms}$ via direction $d \in E$, which the latent policy $\bar{\pi}_{\mathcal{R},d} \in \Pi$ is trained to optimize. To enable high-level reasoning when the rooms are aggregated into a unified model, we add the following assumption.

Assumption 12: Shared reset

All rooms $\mathcal{R} \in \text{Rooms}$ share the same reset state s_{reset} in $\mathcal{H}$.

3.1 Planner Synthesis

Fix Π as a collection of low-level, latent policies. In this section, we show that synthesizing a high-level planner reduces to constructing a policy in a compact model, where the action space coincides with the edges of the graph $\mathcal{G}$ (the choices of the high-level planner). In the following, we describe the chain of reductions leading to this result.

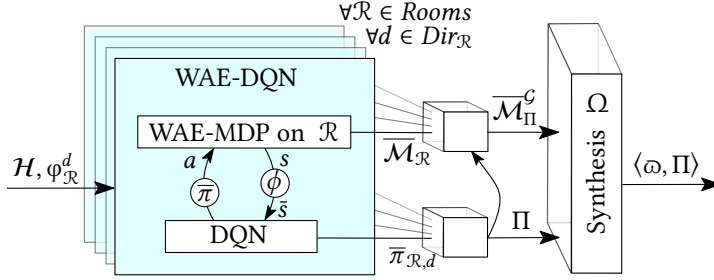


Figure 5.3: Given the hierarchical model $\mathcal{H}$, the set of low-level objectives $\{\varphi_{\mathcal{R}}^d : \mathcal{R} \in \text{Rooms}, d \in \text{Dir}_{\mathcal{R}}\}$, WAE-DQN is run in each room $\mathcal{R} \in \text{Rooms}$ for each direction $d \in \text{Dir}_{\mathcal{R}}$ in parallel. The algorithm outputs an embedding function ϕ , a collection of latent MDPs, and policies Π accompanied by PAC guarantees (Theorem 25). A planner ω maximizing the values of Ω is then synthesized in a *compact model* $\overline{\mathcal{M}}_{\Pi}^{\mathcal{G}}$, aggregated according to the hierarchical graph structure $\mathcal{G}$ of $\mathcal{H}$.

We further discuss the memory requirements of the planner. An overview of this chain of reductions is depicted in Figure 5.4.

We start by establishing a bound on the memory required to implement an optimal planner. We show that *planners indeed require memory*.

Example 20: Memory is necessary for high-level planners

A planner for the hierarchical model depicted in Figure 5.2 requires memory. To reach v_3 while avoiding the bad state s_B , the planner's decision in room $\mathcal{R} = \ell(u)$ must depend on how u is entered: when entering from v_1 , the planner must choose $\leftarrow$, when entering from v_2 , it must choose $\downarrow$, and when entering from v_4 , it must choose $\rightarrow$.

MDP Plan. We define an MDP $\mathcal{M}_{\Pi}$, called an *MDP plan*, such that policies in $\mathcal{M}_{\Pi}$ correspond to planners. Recall that the actions of a planner ω consist of choosing a low-level policy once entering a room. Example 20 shows that the planner needs to take into account the direction from which a room is entered. Accordingly, we define the MDP plan as the tuple $\mathcal{M}_{\Pi} = \langle \mathcal{S}_{\Pi}, \mathcal{A}_{\Pi}, \mathbf{P}_{\Pi}, \mathbf{I}_{\Pi}, \gamma \rangle$, whose components are intuitively defined as follows.

States in $\mathcal{S}_{\Pi}$ keep track of the location of the agent in each room as well as the target of the current low-level policy that is being executed. Precisely, such states have the form $s^* = \langle s, v, u \rangle$, meaning that the current state of $\mathcal{H}$ is $s \in \mathcal{S}_{\mathcal{R}}$, located in the room $\mathcal{R} = \ell(v)$, while the current low-level policy executed is $\bar{\pi}_{\mathcal{R}, d=\langle v, u \rangle}$. The initial distribution

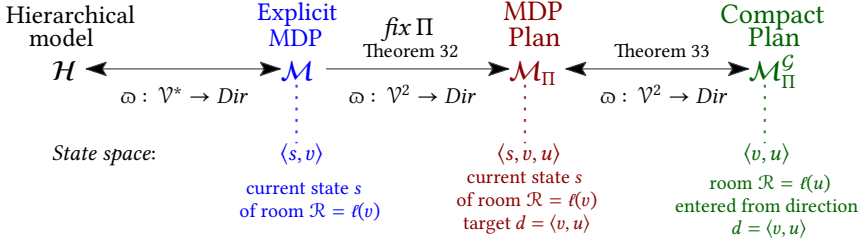


Figure 5.4: Chain of reductions allowing to synthesize a high-level planner ω for a hierarchical model $\mathcal{H}$. The hierarchical model $\mathcal{H}$ can be formulated as an explicit MDP $\mathcal{M}$. Once the low-level latent policies Π are learned (Figure 5.3), the synthesis problem reduces to constructing a stationary policy in an *MDP plan* $\mathcal{M}_{\Pi}$ where the collection Π is fixed, encoding the directions chosen in each room into the state space (Theorem 32). From such a policy, one can derive a $|\mathcal{V}|$ -memory high-level planner ω for $\mathcal{H}$. Finally, finding a policy in the MDP plan is equivalent to finding a policy in a *compact model* $\mathcal{M}_{\Pi}^{\mathcal{G}}$ whose state space merely corresponds to the directions from which rooms are entered, the actions to the high-level choices of the planner, and the transition probabilities to the values achieved by the latent policy chosen in the current room (Theorem 33).

$\mathbf{I}_{\Pi}$ distributes states of the initial room $\mathcal{R}_0 = \ell(v_0)$ according to the entrance function $\mathcal{I}_{\mathcal{R}_0}(\cdot | d_0)$. The action space embeds the choices of the high-level planner: $\mathcal{E} \subseteq \mathcal{A}_{\Pi}$, where $\mathcal{E}$ are the edges of the graph $\mathcal{G}$. The transition function $\mathbf{P}_{\Pi}$ is defined as follows. If s is not an exit state, $\mathbf{P}_{\Pi}(\cdot | s^*)$ follows $\mathbf{P}_{\mathcal{R}}(\cdot | s, a)$, where the action $a \sim \bar{\pi}_{\mathcal{R}, d}(\cdot | s)$ is prescribed by the low-level policy. When s is an exit state, the planner need select a new target direction $d' \in \text{Dir}_{\mathcal{R}'}$ for the next room $\mathcal{R}' = \ell(u)$, and $\mathbf{P}_{\Pi}(\cdot | s^*, d')$ follows the entrance function $\mathcal{I}_{\mathcal{R}'}(\cdot | d)$ of $\mathcal{R}'$ entered from direction d .

Explicit construction of the Plan. Formally, the state space of the MDP plan is

$$\mathcal{S}_{\Pi} = \left(\bigcup_{\mathcal{R} \in \text{Rooms}} (\mathcal{S}_{\mathcal{R}} \setminus \{s_{\text{reset}}\}) \times \mathcal{E} \right) \cup \{s_{\text{reset}}, \perp\},$$

where a pair $\langle s, v, u \rangle \in \mathcal{S}_{\Pi}$ means that the current room is v , the target of the low-level policy is to exit the room in direction $d = \langle v, u \rangle$, and the current state is $s \in \mathcal{S}_{\ell(v)}$. Following Assumption 12, the rooms share the reset state s_{reset} , and $\perp$ is a special sink state that we add for technical reasons, to disable actions in states. The initial distribution $\mathbf{I}_{\Pi}$ has for support $\{\langle s, v, u \rangle \in \mathcal{S}_{\Pi} \mid v = v_0 \text{ and } \langle v, u \rangle = d_1\}$ where states $s \in \mathcal{S}_{\ell(v_0)}$ are distributed according to $\mathcal{I}_{\ell(v_0)}(\cdot | d_0)$. Actions chosen correspond to those of the planner — only required when entering a room — so the action space is $\mathcal{A}_{\Pi} = \mathcal{E} \cup \{*\}$, where

$d \in \mathcal{E}$ means that the low-level policy that is executed exits via direction d , and $*$ is a special action that is used inside a room, indicating no change to the low-level policy. Let $\mathbf{P}$ be the transition function of the explicit MDP $\mathcal{M}$ (cf. Section 1.1). Note that once d is chosen, we only allow exiting the room through direction d .

We now formally define the transition function. For a state $\langle s, v, u \rangle \in \mathcal{S}_\Pi$ with $d = \langle v, u \rangle$,

- (i) if s is not an exit state, i.e., $s \notin \mathcal{O}_{\ell(v)}(d)$, then the action is chosen by the low-level policy $\bar{\pi}_{\ell(v),d}$ and the next state is chosen according to the transitions of $\ell(v)$: for every $s' \in \mathcal{S}_{\ell(v)} \setminus \{s_{reset}\}$,

$$\mathbf{P}_\Pi(\langle s', v, u \rangle \mid \langle s, v, u \rangle, *) = \mathbb{E}_{a \sim \pi_{\ell(v),d}(\cdot \mid s)} \mathbf{P}(\langle s', v \rangle \mid \langle s, v \rangle, a); \quad (5.4)$$

- (ii) if s is an exit state in direction d , i.e., $s \in \mathcal{O}_{\ell(v)}(d)$, the next room is entered from direction d and the planner needs to choose a new target direction d' : for every $s' \in \mathcal{S}_{\ell(u)} \setminus \{s_{reset}\}$ and edge $d' = \langle u, t \rangle \in \text{out}(u)$:

$$\mathbf{P}_\Pi(\langle s', u, t \rangle \mid \langle s, v, u \rangle, d') = \mathbf{P}(\langle s', u \rangle \mid \langle s, v \rangle, a_{exit}) = \mathcal{I}_{\ell(u)}(s' \mid d) \quad (5.5)$$

- (iii) the reset state is handled exactly as in the explicit model $\mathcal{M}$: $\mathbf{P}_\Pi(s_{reset} \mid \langle s, v, u \rangle, *) = \mathbb{E}_{a \sim \pi_{\ell(v),d}} \mathbf{P}(s_{reset} \mid \langle s, v \rangle, a)$, and $\mathbf{P}_\Pi(\cdot \mid s_{reset}, a) = \mathbf{I}_\Pi$ for any $a \in \mathcal{A}_\Pi$;
- (iv) any other undefined distribution transitions deterministically to the sink state $\perp$ so that $\mathbf{P}_\Pi(\perp \mid \perp, a) = 1$ for any $a \in \mathcal{A}_\Pi$.

Proper policies. We say that a policy π for $\mathcal{M}_\Pi$ is *proper* if the decisions of π ensure to almost surely avoid $\perp$, i.e., $\mathbb{P}_\pi^{\mathcal{M}_\Pi}(\diamond \perp) = 0$. Note that *improper* policies strictly consist of those that prescribe to not follow the low-level policy corresponding to the current objective and do not select a new target direction when exiting.

Property 2: High-level objective in the MDP plan

In $\mathcal{M}_\Pi$, the high-level objective Ω translates to the reach-avoid objective $\varphi(\mathbf{T}, \mathbf{B})$ where $\mathbf{T} = \{\langle s, v, u \rangle \in \mathcal{S}_\Pi \mid v \in G\}$ and

$$\mathbf{B} = \{\langle s, v, u \rangle \in \mathcal{S}_\Pi \mid \ell_{\mathcal{R}}(s) \notin \mathbf{B}_{\mathcal{R}} \text{ with } \mathcal{R} = \ell(v)\}$$

so that $G \subseteq \mathcal{V}$ is the set of goal vertices and $\mathbf{B}_{\mathcal{R}}$ is the set to avoid in $\varphi_{\mathcal{R}}^d$ for any $d \in \text{Dir}_{\mathcal{R}}$. When clear from the context, we may simply write $\varphi(\mathbf{T}, \mathbf{B})$ for φ .^a

^aNote that $\mathbf{T}, \mathbf{B}$ are set of states. We consider the natural labelling of $\mathcal{M}_\Pi$: $\ell_\Pi(s^*) = \{s^*\}$ for all $s^* \in \mathcal{S}_\Pi$.

The following theorem states that a deterministic stationary policy in the MDP plan can be implemented through a planner $\omega : \mathcal{V}^2 \rightarrow \mathcal{E}$ which utilizes a memory remembering the direction from which the current room was entered. Such a planner is said to have a memory of size $|\mathcal{V}|$, as one can implement such memory via a Mealy machine (Property 1) with at most $|\mathcal{V}|$ states.

Lemma 23: Equivalence of policies in the hierarchical model and plan

There is an equivalence between high-level planners with memory of size $|\mathcal{V}|$ in the hierarchical model $\mathcal{H}$ and proper deterministic stationary policies in the MDP plan $\mathcal{M}_\Pi$ that preserves the values of their respective objective, so that the values of the objective Ω in $\mathcal{H}$ and ϕ in $\mathcal{M}_\Pi$ are the same under equivalent planners and policies.

Proof. Let ω be a high-level planner for $\mathcal{H}$ with memory of size $|\mathcal{V}|$. Let us encode ω as a finite Mealy machine (Property 1) whose inputs are graph vertices $\mathcal{V}$ and outputs are directions, i.e., $\omega = \langle \mathcal{Q}, \omega_a, \omega_u, q_0 \rangle$ where $\mathcal{Q}$ is a set of memory states with $|\mathcal{Q}| = |\mathcal{V}|$, $\omega_a : \mathcal{V} \times \mathcal{Q} \rightarrow \mathcal{E}$ is the next action function, $\omega_u : \mathcal{V} \times \mathcal{Q} \times \mathcal{E} \rightarrow \mathcal{Q}$ is the memory update function, and q_0 is the initial memory state.

Let us consider the hierarchical controller $\pi = \langle \omega, \Pi \rangle$ as a policy in the explicit MDP $\mathcal{M}$. Since ω is a high-level planner, we require that

1. $\omega_a(v_0, q_0) = d_1$, and
2. if $\omega_a(v, q) = d$, then $d \in \text{out}(v)$ for any $v \in \mathcal{V}, q \in \mathcal{Q}$.

Intuitively, ω_a chooses the direction to follow in the current room based on the current memory state q , and ω_u describes how to update the memory, based on the current room, the current memory state, and the direction chosen. By definition of π (see Section 1), ω_a is used at each time step in the current room, to know which low-level policy to execute, and ω_u is triggered once an exit state is reached, to switch to the next memory state that will determine the direction to follow in the next room.

Notice that the probability measure induced by π in $\mathcal{M}$ is $\mathbb{P}_\pi^{\mathcal{M}}$, which defines a distribution over the trajectories of the Markov chain $\mathcal{M}_\pi$, whose state space is the product of the state space of the explicit MDP, $S = \{ \langle s, v \rangle : s \in S_{\ell(v)}, v \in \mathcal{V} \}$, and the memory states $\mathcal{Q}$, i.e., $S \times \mathcal{Q}$ (cf. Definition 8).

Recall that the σ -algebra of $\mathcal{M}_\pi$ is generated by the countable union, intersection, and complementation of cylinders sets (cf. Remark 1). Basic events are cylinders spanned by finite trajectory prefixes. In particular, in $\mathcal{M}_\pi$, for any given finite trajectory $\tau = \langle s_{0:t}, v_{0:t}, q_{0:t} \rangle \in \text{Traj}_{\mathcal{M}_\pi}^{\text{fin}}$, the cylinder spanned by τ is given by

$$\text{Cyl}(\tau) = \{ \langle s'_{0:\infty}, v_{0:\infty}, q'_{0:\infty} \rangle \in \text{Traj}_{\mathcal{M}_\pi} \mid s'_n = s_n, v'_n = v_n, \text{ and } q'_n = q_n \text{ for all } n \leq t \}.$$

Following the definitions of the induced MC $\mathcal{M}_\pi$ (Definition 8), the probability space $\mathbb{P}_\pi^{\mathcal{M}}$ (Remark 1), and the controller π (cf. Section 1.1), the measure $\mathbb{P}_\pi^{\mathcal{M}}$ can be obtained

inductively as follows. For a state $\langle s, v \rangle \in \mathcal{S}$, $\mathbb{P}_\pi^{\mathcal{M}}(\text{Cyl}(s, v, q)) = \mathcal{I}_{\ell(v_0)}(s \mid d_0)$ if $v_0 = v$ and $q = q_0$, and assigns a zero probability otherwise. The probability of a cylinder spanned by a trajectory $\tau = \langle s_{0:t}, v_{0:t}, q_{0:t} \rangle$ is given as follows:

- (a) if s_{t-1} is not an exit state, the low-level policy is executed in direction $d = \omega_a(v_{t-1}, q_{t-1})$ and both the current vertex and memory state must remain unchanged:

$$\mathbb{P}_\pi^{\mathcal{M}}(\text{Cyl}(s_{0:t-1}, v_{0:t-1}, q_{0:t-1})) \cdot \mathbb{E}_{a \sim \bar{\pi}_{\ell(v_t), d}(\cdot \mid s)} \mathbf{P}(\langle s_t, v_t \rangle \mid \langle s_{t-1}, v_{t-1} \rangle, a)$$

if $s_{t-1} \notin \mathcal{O}_{\ell(v_{t-1})}(d)$ with $d = \omega_a(v_{t-1}, q_{t-1})$, $v_t = v_{t-1}$, and $q_t = q_{t-1}$;

- (b) if s_{t-1} is an exit state in the direction prescribed in q_{t-1} , then this direction should point to v_t and the memory state must be updated to q_t :

$$\mathbb{P}_\pi^{\mathcal{M}}(\text{Cyl}(s_{0:t-1}, v_{0:t-1}, q_{0:t-1})) \cdot \mathcal{I}_{\ell(v_t)}(s_t \mid d)$$

if $s_{t-1} \in \mathcal{O}_{\ell(v_{t-1})}(d)$ with $d = \omega_a(v_{t-1}, q_{t-1}) = \langle v_{t-1}, v_t \rangle$, and $q_t = \omega_u(v_{t-1}, q_{t-1}, d)$;

- (c) if s_{t-1} is the reset state, by Assumptions 6 and 12, the planner must be reset as well:

$$\mathbb{P}_\pi^{\mathcal{M}}(\text{Cyl}(s_{0:t-1}, v_{0:t-1}, q_{0:t-1})) \cdot \mathcal{I}_{\ell(v_0)}(s_t \mid d_0)$$

if $s_{t-1} = s_{\text{reset}}$, $v_t = v_0$, and $q_t = q_0$;

- (d) zero otherwise.

Notice that renaming $\mathcal{Q}$ to $\mathcal{V}$ so that for all $q \in \mathcal{Q}$, q is changed to $u \in \mathcal{V}$ (i.e., $q \mapsto u$) whenever $\omega_a(v, q) = \langle v, u \rangle$ is harmless, since the probability measure remains unchanged. From now on, we consider that $\mathcal{Q}$ has been renamed to $\mathcal{V}$ in this manner. Now, define the relation $\equiv$ between planners in $\mathcal{M}$ and policies in $\mathcal{M}_\Pi$ as

$\omega \equiv \pi$ if and only if

$$\pi(\langle s, v, u \rangle) = \begin{cases} \omega(v \cdot u) & \text{if } s \in \mathcal{O}_{\ell(v)}(\langle v, u \rangle) \\ * & \text{otherwise,} \end{cases}$$

for all $\langle s, v, u \rangle \in \mathcal{S}_\Pi$, where we write $\omega(v \cdot u)$ for $\omega_a(u, \cdot) \circ \omega_u(v, u, d = \cdot) \circ \omega_a(v, u)$. By construction of $\mathcal{M}_\Pi$, modulo the renaming of $\mathcal{Q}$ to $\mathcal{V}$, $\mathbb{P}_{\pi=\langle \omega, \Pi \rangle}^{\mathcal{M}} = \mathbb{P}_\pi^{\mathcal{M}_\Pi}$ for any ω, π in relation $\omega \equiv \pi$: condition (i) is equivalent to (a), condition (ii) is equivalent to (b), and condition (iii) is equivalent to (c). Note that the only policies π which cannot be in relation with some planner ω are *improper policies*, i.e., those choosing actions leading to the sink state $\perp$ (see condition iv). Such policies are discarded by assumption.

The result follows from the fact that, modulo the renaming of $\mathcal{Q}$ to $\mathcal{V}$, planners and policies in relation $\equiv$ lead to the same probability space. $\square$

Theorem 32: Tight memory bound for the planner

For a fixed collection of low-level policies Π , a memory of size $|\mathcal{V}|$ is necessary and sufficient for the high-level planner to maximize the values of Ω in the hierarchical model $\mathcal{H}$.

Proof. The necessity of a memory of size $|\mathcal{V}|$ is shown in Example 20. The sufficiency follows from the fact that optimal deterministic stationary policies are sufficient to maximize reach-avoid objectives in MDPs.

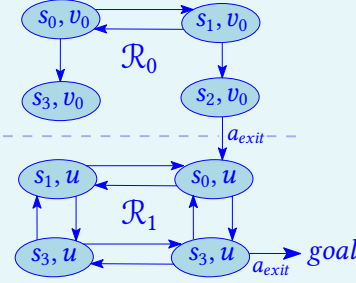
To see why, consider the reach-avoid objective φ in the MDP plan $\mathcal{M}_\Pi$. The optimal value of φ is given as follows, for every state $s^* \in S_\Pi$:

$$V^*(s, \varphi) = \begin{cases} 1 & \text{if } s \in \mathbf{T} \setminus \mathbf{B}, \\ 0 & \text{if } s \in \mathbf{B}, \\ \gamma \cdot \max_{a \in \mathcal{A}_\Pi} \mathbb{E}_{s' \sim P_\Pi(\cdot | s^*, a)} V^*(s', \varphi) & \text{otherwise} \end{cases} \quad (5.6)$$

(cf. Chapter 2, Section 3.6). This value function is actually equivalent to the optimal value function of the usual expected return in an MDP where we make states from $\mathbf{T} \cup \mathbf{B}$ absorbing and incur a reward of $(1 - \gamma)$ for any action played in the absorbing states $\mathbf{T} \setminus \mathbf{B}$ (leading to a value of 1 in those states). Then, by Theorem 5, deterministic stationary policies are sufficient.

Let π^* be a proper optimal deterministic stationary policy in $\mathcal{M}_\Pi$. Note that one can always find a proper optimal policy from an improper one: if π^* is improper, it is necessarily because a prohibited action has been chosen after having reached the target, which can be replaced by any other action without changing the value of the objective. Consider a planner ϖ in the hierarchical model $\mathcal{H}$ which is equivalent to π^* (Lemma 23). Then, ϖ is optimal for Ω in $\mathcal{H}$ (since the probability space of the two models is the same), and ϖ uses a memory of size $|\mathcal{V}|$. $\square$

Compact Plan. We now describe a *compact* construction of $\mathcal{M}_\Pi$, denoted $\mathcal{M}_\Pi^{\mathcal{C}}$, that preserves its value function. The construction depends on analysis of the rooms under control of the low-level policies. In the next section, we show how to estimate this value from the latent models. Intuitively, each state of $\mathcal{M}_\Pi^{\mathcal{C}}$ consists of a pair of vertices $\langle v, u \rangle$ indicating that room $\mathcal{R} = \ell(u)$ is entered from direction $d = \langle v, u \rangle$. As in $\mathcal{M}_\Pi$, a planner must select an exit direction $\varpi(v \cdot u) = d' = \langle u, v' \rangle$ for room $\mathcal{R}$. We set the probability of going to v' by choosing d' to the value that the low-level policy $\bar{\pi}_{\mathcal{R}, d'}$ achieves in room $\mathcal{R}$. We illustrate the idea behind the construction via the following example.

Example 21: Composed trajectories

Consider the explicit model of Figure 5.1, which we project on two dimensions in the left figure. Each directed arrow corresponds to a transition with a non-zero probability. A state of the form $\langle s, v \rangle$ indicates that the agent is in state s of room $\ell(v)$.

Consider a trajectory τ that enters $\ell(v_0) = \mathcal{R}_0$, exits after $i = 3$ steps ($s_0 \rightarrow s_1 \rightarrow s_2 \xrightarrow{a_{exit}}$), enters $\ell(u) = \mathcal{R}_1$, exits after $j = 2$ steps ($s_0 \rightarrow s_3 \xrightarrow{a_{exit}}$), and reaches the high-level goal. The prefix of τ in $\mathcal{R}_0$ is discounted to γ^3 when the agent exits.

Similarly, the suffix of τ in $\mathcal{R}_1$ is discounted to γ^2 . Once in the goal, the agent gets a “reward” of one (the goal is reached). The discounted reward of τ is thus $\gamma^{i+j} = \gamma^5$. In expectation, this corresponds to multiplying the values in the individual rooms and, in turn, with the semantics of $\mathcal{M}_{\Pi}^G$ where probabilities are multiplied along a trajectory.

Formally, $\mathcal{M}_{\Pi}^G$ is an MDP $\langle S, \mathcal{A}, \mathbf{P}, \mathbf{AP} = \{\text{goal}\}, \ell^*, \mathbf{I} \rangle$, where $S = \mathcal{E} \cup \{\perp\}$, $\mathcal{A} = \mathcal{E}$, $\mathbf{I}(d_0) = 1$, and $\ell^*(\langle v, u \rangle) = \{\text{goal}\}$ if and only if $u \in G$ (the set of goal vertices). The transition function is defined as follows. Let $u \in \mathcal{V}$, for entrance and exit directions $\langle v, u \rangle, \langle u, t \rangle \in \text{out}(u)$, the probability of transitioning from $\langle v, u \rangle$ to $\langle u, t \rangle$ is

$$\mathbf{P}(d' = \langle u, t \rangle \mid \langle v, u \rangle, d = \langle u, t \rangle) = \mathbb{E}_{s \sim \mathbf{I}_{\ell(u)}(\cdot \mid \langle v, u \rangle)} V^{\bar{\pi}_{\ell(u), d}}(s, \varphi_{\ell(u)}^d). \quad (5.7)$$

The remaining probability mass is concentrated on a special sink state $\perp$, capturing the case where the low-level policies fail to satisfy their objective ($\neg \varphi_{\ell(u)}^d$):

$$\mathbf{P}(\perp \mid \langle v, u \rangle, d = \langle u, t \rangle) = 1 - \mathbf{P}(\langle u, t \rangle \mid \langle v, u \rangle, d = \langle u, t \rangle).$$

Precisely, let $v, u \in \mathcal{V}$, $d \in \mathcal{E}$, and $d' \in \mathcal{E} \cup \{\perp\}$, $\mathbf{P}$ is explicitly defined as

$$\mathbf{P}(d' \mid \langle v, u \rangle, d) = \begin{cases} \mathbb{E}_{s \sim \mathbf{I}_{\ell(u)}(\cdot \mid \langle v, u \rangle)} V^{\bar{\pi}_{\ell(u), d}}(s, \varphi_{\ell(u)}^d) & \text{if } d = d' \in \text{out}(u), \\ 1 - \mathbf{P}(d \mid \langle v, u \rangle, d) & \text{if } d' = \perp \text{ and } d \in \text{out}(u), \\ 1 & \text{if } d' = \perp \text{ and } d \notin \text{out}(u), \text{ and} \\ 0 & \text{otherwise,} \end{cases} \quad (5.8)$$

while $P(\perp \mid \perp, d) = 1$.

For convenience, we assume that $\text{reset} \in \mathbf{B}_{\mathcal{R}}$ for each room $\mathcal{R}$, which is consistent with Remark 29. The construction of $\mathcal{M}_{\Pi}^G$ and Theorem 33 can be generalized by additionally wisely handling the reset state in $\mathbf{P}$.

Theorem 33: Value equality in the compact plan

Let $\pi = \langle \omega, \Pi \rangle$ be a hierarchical controller for $\mathcal{H}$ with a $|\mathcal{V}|$ -memory planner ω . Denote by $V_{\mathcal{M}_{\Pi}}^{\pi}(\varphi)$ the initial value of $\mathcal{M}_{\Pi}$ running under a policy π equivalent^a to ω in $\mathcal{M}_{\Pi}$ for the reach-avoid objective φ of Property 2. Moreover, denote by $V_{\mathcal{M}_{\Pi}^G}^{\omega}(\diamond^Y \text{goal})$ the initial value obtained in $\mathcal{M}_{\Pi}^G$ when the agent follows the decisions of ω for the discounted reachability objective $\diamond^Y \text{goal}$ (cf. Chapter 2, Section 3.6). Then, assuming $v_0 \notin G$ (the case where $v_0 \in G$ is trivial),

$$V_{\mathcal{M}_{\Pi}}^{\pi}(\varphi) = V_{\mathcal{M}_{\Pi}^G}^{\omega}(\diamond^Y \text{goal}).$$

^acf. Lemma 23.

Proof. Let $\tilde{\mathcal{M}} = \langle \tilde{\mathcal{S}}, \tilde{\mathcal{A}}, \tilde{\mathbf{P}}, \tilde{\mathbf{I}}, \tilde{\mathbf{A}\mathbf{P}}, \tilde{\ell}, \gamma \rangle$ be any MDP, $\tilde{\pi}$ be a stationary policy for $\tilde{\mathcal{M}}$, and $\varphi(\mathbf{T}, \mathbf{B})$ be a reach-avoid objective with $\mathbf{T}, \mathbf{B} \subseteq \tilde{\mathbf{A}\mathbf{P}}$. The value of $\varphi(\mathbf{T}, \mathbf{B})$ in $\tilde{\mathcal{M}}$ is given by

$$V^{\tilde{\pi}}(s, \varphi) = \begin{cases} 1 & \text{if } \tilde{\ell}(s) \cap \mathbf{T} \neq \emptyset \text{ and } \tilde{\ell}(s) \cap \mathbf{B} = \emptyset, \\ 0 & \text{if } \tilde{\ell}(s) \cap \mathbf{B} \neq \emptyset, \text{ and} \\ \gamma \cdot \mathbb{E}_{a \sim \tilde{\pi}(\cdot|s)} \mathbb{E}_{s' \sim \tilde{\mathbf{P}}(\cdot|s, a)} V^{\tilde{\pi}}(s', \varphi) & \text{otherwise.} \end{cases}$$

(cf. Chapter 2, Section 3.6). Now, let us consider the values of the MDP plan $\mathcal{M}_{\Pi}$ for the reach-avoid objective $\varphi(\mathbf{T}, \mathbf{B})$ of Property 2. Fix a $|\mathcal{V}|$ -memory planner ω and let π be an equivalent policy for $\mathcal{M}_{\Pi}$ (Lemma 23). We take a close look at the value of each state in $\mathcal{M}_{\Pi}$ by following the same structure as we used for defining the transition function of $\mathcal{M}_{\Pi}$ (cf. Section 3.1). For the sake of presentation and by abuse of notation, given any pair of vertices $v, u \in \mathcal{V}$, we use $\langle s_{\text{reset}}, v, u \rangle$ to refer to the (unified, cf. Assumption 12) reset state $s_{\text{reset}} \in S_{\Pi}$. Then, given a state $\langle s, v, u \rangle \in S_{\Pi}$ with direction $d = \langle v, u \rangle$,

- (i) if s is not an exit state, i.e., if $s \notin \mathcal{O}_{\ell(v)}(d)$, then

$$\begin{aligned} & V^{\pi}(\langle s, v, u \rangle, \varphi) \\ &= \gamma \cdot \mathbb{E}_{\langle s', v, u \rangle \sim \mathbf{P}_{\Pi}(\cdot | \langle s, v, u \rangle, *)} V^{\pi}(\langle s', v, u \rangle, \varphi) \quad (\text{by Equation 5.4}) \\ &= \gamma \cdot \sum_{s' \in S_{\ell(v)}} \mathbf{P}_{\Pi}(\langle s', v, u \rangle \mid \langle s, v, u \rangle, *) \cdot V^{\pi}(\langle s', v, u \rangle, \varphi) \end{aligned}$$

$$= \gamma \cdot \sum_{s' \in \mathcal{S}_{\ell(v)}} \sum_{a \in \mathcal{A}_{\ell(v)}} \bar{\pi}_{\ell(v),d}(a | s) \cdot \mathbf{P}_{\ell(v)}(s' | s, a) \cdot V^\pi(\langle s', v, u \rangle, \varphi);$$

- (ii) if s is an exit state in the direction d , i.e., $s \in \mathcal{O}_{\ell(v)}(d)$, given the direction chosen by the planner $d' = \omega(v \cdot u) = \langle u, t \rangle \in \text{out}(u)$, we have

$$\begin{aligned} & V^\pi(\langle s, v, u \rangle, \varphi) \\ &= \gamma \cdot \mathbb{E}_{\langle s', u, t \rangle \sim \mathbf{P}_{\Pi}(\cdot | \langle s, v, u \rangle, d')} V^\pi(\langle s', u, t \rangle, \varphi) \\ &= \gamma \cdot \mathbb{E}_{s' \sim \mathcal{I}_{\ell(u)}(\cdot | d)} V^\pi(\langle s', u, t \rangle, \varphi) \quad (\text{by Equation 5.5}) \\ &= \gamma \cdot \sum_{s' \in \mathcal{S}_{\ell(u)}} \mathcal{I}_{\ell(u)}(s' | d) \cdot V^\pi(\langle s', u, t \rangle, \varphi); \end{aligned} \quad (5.9)$$

- (iii) if v is a goal, i.e., $v \in G$, $V^\pi(s, \varphi) = 1$; and

- (iv) otherwise, when s is a bad state, i.e., $\ell_{\mathcal{R}}(s) \cap \mathbf{B}_{\ell(v)} \neq \emptyset$, $V^\pi(s, \varphi) = 0$.

Take $\mathcal{R} = \ell(v)$. By (i), (ii), and by definition of the probability measure $\mathbb{P}_{\pi_{\mathcal{R},d}}^{\mathcal{R}}$ (Theorem 1 and Remark 1), when s is not an exit state, i.e., $s \notin \mathcal{O}_{\ell(v)}(d)$, we have

$$V^\pi(\langle s, v, u \rangle, \varphi) = \sum_{\langle s_0:i, a_0:i-1 \rangle \in \text{Traj}_{\varphi_{\mathcal{R}}^d}^{\text{fin}}} \gamma^i \cdot \mathbb{P}_{\pi_{\mathcal{R},d}}^{\mathcal{R}}(\text{Cyl}(s_0:i, a_0:i-1) | s_0 = s) \cdot V^\pi(\langle s_i, v, u \rangle, \varphi),$$

where

$$\text{Traj}_{\varphi_{\mathcal{R}}^d}^{\text{fin}} = \left\{ \tau = \langle s_0:i, a_0:i-1 \rangle \in \text{Traj}_{\mathcal{R}}^{\text{fin}} \mid s_i \in \mathcal{O}_{\mathcal{R}}(d) \text{ and } \forall j \leq i, \ell_{\mathcal{R}}(s_j) \cap \mathbf{B}_{\mathcal{R}} = \emptyset \right\}.$$

Following Equation 5.9, notice that $V^\pi(\langle s_{\text{exit}}, v, u \rangle, \varphi) = V^\pi(\langle s'_{\text{exit}}, v, u \rangle, \varphi)$ for any $s_{\text{exit}}, s'_{\text{exit}} \in \mathcal{O}_{\mathcal{R}}(d = \langle v, u \rangle)$: the probability of going to the next room $\mathcal{R}' = \ell(u)$ from an exit state of the current room $\mathcal{R}$ only depends on the entrance function $\mathcal{I}_{\mathcal{R}'}$ and is independent from the exact exit state which allowed to leave the current room $\mathcal{R}$.

Let us denote by $V^\pi(\langle \cdot, v, u \rangle, \varphi)$ the value of any exit state of $\mathcal{R}$, i.e., $V^\pi(\langle \cdot, v, u \rangle, \varphi) = V^\pi(\langle s_{\text{exit}}, v, u \rangle, \varphi)$ for all $s_{\text{exit}} \in \mathcal{O}_{\mathcal{R}}(d)$. Then, we have

$$\begin{aligned} & V^\pi(\langle s, v, u \rangle, \varphi) \\ &= \sum_{\langle s_0:i, a_0:i-1 \rangle \in \text{Traj}_{\varphi_{\mathcal{R}}^d}^{\text{fin}}} \gamma^i \cdot \mathbb{P}_{\pi_{\mathcal{R},d}}^{\mathcal{R}}(\text{Cyl}(s_0:i, a_0:i-1) | s_0 = s) \cdot V^\pi(\langle s_i, v, u \rangle, \varphi) \\ &= \sum_{\langle s_0:i, a_0:i-1 \rangle \in \text{Traj}_{\varphi_{\mathcal{R}}^d}^{\text{fin}}} \gamma^i \cdot \mathbb{P}_{\pi_{\mathcal{R},d}}^{\mathcal{R}}(\text{Cyl}(s_0:i, a_0:i-1) | s_0 = s) \cdot V^\pi(\langle \cdot, v, u \rangle, \varphi) \end{aligned}$$

$$\begin{aligned}
 &= V^\pi(\langle \cdot, v, u \rangle, \varphi) \cdot \sum_{\langle s_0:i, a_0:i-1 \rangle \in \text{Tra}_{\varphi_{\mathcal{R}}^d}^{\text{fin}}} \gamma^i \cdot \mathbb{P}_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}}(\text{Cyl}(s_0:i, a_0:i-1) \mid s_0 = s) \\
 &= V^\pi(\langle \cdot, v, u \rangle, \varphi) \cdot \mathbb{E}_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}} \left[\sup_{i \geq 0} \gamma^i \cdot \mathbb{1} \left\{ s_i \in \mathcal{O}_{\mathcal{R}}(d) \text{ and } \forall j \leq i, \ell_{\mathcal{R}}(s_j) \cap \mathbf{B}_{\mathcal{R}} = \emptyset \right\} \mid s_0 = s \right] \\
 &= V^\pi(\langle \cdot, v, u \rangle, \varphi) \cdot \mathbb{E}_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}} \left[\mathbb{O} \varphi_{\mathcal{R}}^d \mid s_0 = s \right] \quad (\text{see Theorem 13}) \\
 &= V^\pi(\langle \cdot, v, u \rangle, \varphi) \cdot V^{\bar{\pi}_{d,\mathcal{R}}} \left(s, \varphi_{\mathcal{R}}^d \right),
 \end{aligned}$$

where $V^{\bar{\pi}_{d,\mathcal{R}}}(s, \varphi_{\mathcal{R}}^d)$ denotes the value of the reach-avoid objective $\varphi_{\mathcal{R}}^d$ in the room $\mathcal{R}$ from state $s \in \mathcal{R}$ in direction $d = \langle v, u \rangle$. Then, by (ii), assuming $v \notin G$, we have

1. if $u \notin G$,

$$\begin{aligned}
 &V^\pi(\langle \cdot, v, u \rangle, \varphi) \\
 &= \gamma \cdot \sum_{s' \in \mathcal{S}_{\ell(u)}} \mathcal{I}_{\ell(u)}(s' \mid d = \langle v, u \rangle) \cdot V^\pi(\langle s', \varpi(v \cdot u) \rangle, \varphi) \quad (5.10) \\
 &= \gamma \cdot \sum_{s' \in \mathcal{S}_{\ell(u)}} \mathcal{I}_{\ell(u)}(s' \mid d = \langle v, u \rangle) \cdot V^{\bar{\pi}_{\ell(u), \varpi(v \cdot u)}}(s, \varphi_{\mathcal{R}}^d) \cdot V^\pi(\langle \cdot, \varpi(v \cdot u) \rangle, \varphi) \\
 &= \gamma \cdot \mathbf{P}(\varpi(v \cdot u) \mid \langle v, u \rangle, \varpi(v \cdot u)) \cdot V^\pi(\langle \cdot, \varpi(v \cdot u) \rangle, \varphi) \\
 &\quad (\text{where } \mathbf{P} \text{ is the transition function of } \mathcal{M}_{\Pi}^{\mathcal{G}}, \text{ see Equation 5.8})
 \end{aligned}$$

2. if $u \in G$,

$$\begin{aligned}
 &V^\pi(\langle \cdot, v, u \rangle, \varphi) \\
 &= \gamma \cdot \sum_{s' \in \mathcal{S}_{\ell(u)}} \mathcal{I}_{\ell(u)}(s' \mid d = \langle v, u \rangle) \cdot V^\pi(\langle s', \varpi(v \cdot u) \rangle, \varphi) \\
 &= \gamma \cdot \sum_{s' \in \mathcal{S}_{\ell(u)}} \mathcal{I}_{\ell(u)}(s' \mid d = \langle v, u \rangle) \cdot 1 \\
 &\quad (\text{since } \varpi(v \cdot u) = \langle u, t \rangle \text{ for some } t \in \text{Neighbors}(u)) \\
 &= \gamma.
 \end{aligned}$$

Now, respectively denote by $V_{\mathcal{M}_{\Pi}}^\pi(\cdot, \varphi) := V^\pi(\cdot, \varphi)$ and $V_{\mathcal{M}_{\Pi}}^\varpi(\cdot, \diamond^Y \text{goal})$ the value functions of $\mathcal{M}_{\Pi}$ and $\mathcal{M}_{\Pi}^{\mathcal{G}}$ for the objectives φ and $\diamond^Y \text{goal}$. By 1 and 2, and by construction of $\mathcal{M}_{\Pi}^{\mathcal{G}}$, we have for any pair of vertices $v, u \in \mathcal{V}$ that

$$V_{\mathcal{M}_{\Pi}}^\pi(\langle \cdot, v, u \rangle, \varphi) = \gamma \cdot V_{\mathcal{M}_{\Pi}}^\varpi(\langle v, u \rangle, \diamond^Y \text{goal}),$$

On the one hand, notice that, by construction of $\mathcal{M}_{\Pi}^G$, we have for any pair of vertices $\langle v, u \rangle \in \mathcal{E}$ that the initial values $V_{\mathcal{M}_{\Pi}^G}^{\omega}(\diamond^Y \text{goal})$ are $\mathbb{E}_{d \sim \mathbf{I}} V_{\mathcal{M}_{\Pi}^G}^{\omega}(d, \diamond^Y \text{goal}) = V_{\mathcal{M}_{\Pi}^G}^{\omega}(d_0, \diamond^Y \text{goal})$. On the other hand, we have

$$V_{\mathcal{M}_{\Pi}}^{\pi}(\varphi) = \mathbb{E}_{s' \sim \mathbf{I}_{\ell(v_0)}(\cdot | d_0)} V_{\mathcal{M}_{\Pi}}^{\pi}(\langle s', \omega(d_0) \rangle, \varphi) = 1/\gamma \cdot V_{\mathcal{M}_{\Pi}}^{\pi}(\cdot, d_0, \varphi). \quad (\text{by Equation 5.10})$$

Then, we finally have:

$$V_{\mathcal{M}_{\Pi}}^{\pi}(\varphi) = 1/\gamma \cdot V_{\mathcal{M}_{\Pi}}^{\pi}(\cdot, d_0, \varphi) = \gamma/\gamma \cdot V_{\mathcal{M}_{\Pi}^G}^{\omega}(d_0, \diamond^Y \text{goal}) = V_{\mathcal{M}_{\Pi}^G}^{\omega}(\diamond^Y \text{goal}),$$

which concludes the proof. $\square$

Synthesis. Note that the values $V^{\pi_{\mathcal{R}, \cdot}}$ of each room $\mathcal{R} \in \text{Rooms}$ in Equation 5.7 are either unknown (their model is not necessarily explicitly given) or computationally intractable (due to large state spaces). However, assuming access to a latent model for each room, and assuming Π is solely composed of latent policies, we can compute the *latent* value of each low-level objective using standard verification techniques for discounted reachability objectives (cf. Chapter 2, Section 3.8). Assuming we are given a *latent entrance function* for each room $\mathcal{R} \in \text{Rooms}$, we may construct a *compact latent MDP plan* $\overline{\mathcal{M}}_{\Pi}^G$ in a similar fashion as the original one. This enables the computation of the value of the global objective function $\bar{V}_{\mathbf{I}}^{\omega}(\diamond^Y \text{goal})$ for any hierarchical controller $\pi = \langle \omega, \Pi \rangle$. Standard techniques for discounted reachability in MDPs further allow us to synthesize a planner for $\overline{\mathcal{M}}_{\Pi}^G$. Since the state space of $\overline{\mathcal{M}}_{\Pi}^G$ is identical to that of $\mathcal{M}_{\Pi}^G$, planners for $\overline{\mathcal{M}}_{\Pi}^G$ can be executed in $\mathcal{M}_{\Pi}^G$ and vice versa.

3.2 Reusable Latent Policies

To learn one latent model per room and the low-level policies Π , we run WAE-DQN in each room $\mathcal{R}$ independently (Figure 5.1). Viewing $\mathcal{R}$ as an MDP, we obtain a transition loss $L_{\mathbf{P}}^{\mathcal{R}, d}$ for each directions d associated with a latent policy $\bar{\pi}_{\mathcal{R}, d} \in \Pi$. Nevertheless, independent training introduces some complications. Room $\mathcal{R}$ has its own initial distribution $\mathbf{I}_{\mathcal{R}}$, while, at synthesis time, the initial distribution depends on controller $\pi = \langle \omega, \Pi \rangle$ and is a marginalization of $\mathbf{I}_{\mathcal{R}}(\cdot | d)$ according to the directions d chosen by ω . Recall that $L_{\mathbf{P}}^{\mathcal{R}, d}$ is the total variation between original and latent transition functions, marginalized with respect to the states produced under $\xi_{\bar{\pi}_{\mathcal{R}, d}}$, i.e., the distributions over states visited under $\bar{\pi}$ when using $\mathbf{I}_{\mathcal{R}}$ as entrance function, and not $\mathbf{I}_{\mathcal{R}}$. Since $\xi_{\bar{\pi}_{\mathcal{R}, d}}$ could be unrelated to the distribution over states visited under high-level controller π , $L_{\mathbf{P}}^{\mathcal{R}, d}$ (and thus the guarantees from WAE-DQN) could turn obsolete and non-reusable.

Figure 5.5 illustrates this distribution shift. Assume that ϖ chooses direction $\downarrow$ in $\mathcal{R}$. As $\mathbf{I}_{\mathcal{R}}$ is uniform, every state is included in the support of the distribution $\xi_{\bar{\pi}_{\mathcal{R},\downarrow}}$ over states visited under $\bar{\pi}_{\mathcal{R},\downarrow}$ at training time. In contrast, under a high-level controller, rooms are entered according to $\mathbf{I}_{\mathcal{R}}(\cdot \mid d \in \{\downarrow, \leftarrow\})$ (whose supports are depicted in red and green). Since the goal is to go down, all states of $\mathcal{R}$ need not be visited under $\bar{\pi}_{\mathcal{R},\downarrow}$, i.e., the distribution over states visited under the controller need not be the same as during training. The question is whether it is possible to recover the guarantees associated with latent models at synthesis time.

Fortunately, as we will show in the following proposition, it turns out that if the initial distribution $\mathbf{I}_{\mathcal{R}}$ of each room $\mathcal{R}$ is well designed and provides sufficient coverage of the state space of $\mathcal{R}$, it is possible to learn a *latent entrance function* $\bar{\mathbf{I}}_{\mathcal{R}}$ so that the guarantees associated with each room can be lifted to the high-level controller. To learn this entrance function, we define an additional *entrance loss* as

$$L_{\mathcal{I}} = \frac{1}{2} \mathbb{E}_{\mathcal{R}, d \sim \xi_{\pi}} \left\| \phi \mathbf{I}_{\mathcal{R}}(\cdot \mid d) - \bar{\mathbf{I}}_{\mathcal{R}}(\cdot \mid d) \right\|_1, \quad (5.11)$$

where $\phi \mathbf{I}_{\mathcal{R}}(\cdot \mid d) = \mathbb{E}_{\bar{s} \sim \mathbf{I}(\cdot \mid d)} \mathbb{1}\{\bar{s} = \phi(s)\}$, $\bar{\mathbf{I}}_{\mathcal{R}} : \text{Dir}_{\mathcal{R}} \rightarrow \Delta(\bar{\mathcal{S}})$ is the latent entrance function, π is the stationary policy in $\mathcal{M}_{\Pi}$ corresponding to the hierarchical controller $\pi = \langle \varpi, \Pi \rangle$ in $\mathcal{H}$ where ϖ has a memory of size $|\mathcal{V}|$, and ξ_{π} is the stationary distribution induced by π in $\mathcal{M}_{\Pi}$. The measure ξ_{π} can also be seen as a distribution over rooms and directions chosen under the controller:²

$$\xi_{\pi}(\mathcal{R}, d) = \mathbb{E}_{s, v, u \sim \xi_{\pi}} [\mathbb{1}\{\mathcal{R} = \ell(v), d = \langle v, u \rangle\}].$$

²For simplicity, we consider here the special state $\langle s_{\text{reset}}, v, v_0 \rangle$ with $\langle v, v_0 \rangle = d_0$ as the joint reset state of the model (Assumption 12). This special state is further labelled as **reset**.

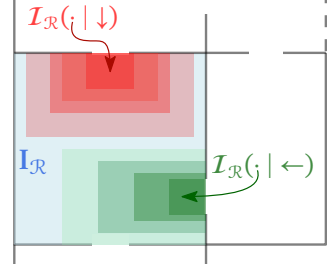


Figure 5.5: Uniform initial distribution $\mathbf{I}_{\mathcal{R}}$ (blue) and entrance function $\mathbf{I}_{\mathcal{R}}$ (red: $\downarrow$, green: $\leftarrow$). Color intensity shows probability mass.

Proposition 2: Reusable RL components

Let $\pi = \langle \omega, \Pi \rangle$ be a hierarchical controller in $\mathcal{H}$ where ω has finite memory of size $|\mathcal{V}|$ and Π only consists of latent policies, and π be the equivalent stationary policy in the MDP plan $\mathcal{M}_\Pi$.

(State coverage) Assume that for any training room $\mathcal{R} \in \text{Rooms}$ and direction $d \in \text{Dir}_{\mathcal{R}}$, the projection of the BSCC associated to π in $\mathcal{M}_\Pi$ to $\mathcal{S}_{\mathcal{R}}$ is included into the BSCC associated with low-level policy $\bar{\pi}_{\mathcal{R},d}$ in $\mathcal{R}$.

Then, there exists a constant $K \geq 0$ so that

$$\left| V_{\mathcal{M}}^{\pi}(\varphi) - \bar{V}_{\bar{\Pi}}^{\omega}(\diamond^{\gamma} \text{goal}) \right| \leq \frac{L_I + K \cdot \mathbb{E}_{\mathcal{R}, d \sim \xi_{\pi}} L_{\mathbf{P}}^{\mathcal{R}, d}}{\xi_{\pi}(s_{\text{reset}}) \cdot (1 - \gamma)}.$$

The state coverage assumption basically boils down to designing the initial distribution of each room so that its support covers the states that are relevant for the high-level controller. Note that the easiest way to achieve this is to cover the whole state space of the room or the joint support of its entrance function for all the possible input directions. In the latter case, one needs to have access to the entrance function of the room, which is rarely the case — this is further the reason why we propose to learn a latent entrance function.

The remaining of this section is dedicated to the proof of Proposition 2. We start by defining the specific training setting of each independent room.

Training rooms. Formally, for each room $\mathcal{R} \in \text{Rooms}$ and direction $d \in \text{Dir}_{\mathcal{R}}$, we train a WAE-DQN agent by considering $\mathcal{R}$ as episodic MDP with *some* initial distribution $\mathbf{I}_{\mathcal{R}}$, yielding a (i) low-level latent policy $\bar{\pi}_{\mathcal{R},d}$, (ii) latent MDP $\bar{\mathcal{M}}_{\mathcal{R}}$, and (iii) state-embedding function $\phi_{\mathcal{R}}$. Since $\bar{\pi}_{\mathcal{R},d}$ must learn to maximize the values of the objective $\varphi_{\mathcal{R}}^d$, which asks to reach the exit state in direction d , we restart the simulation when the latter is visited. Formally, the related training room is an episodic MDP $\mathcal{R}_d = \langle \mathcal{S}_{\mathcal{R}}, \mathcal{A}_{\mathcal{R}}, \mathbf{P}_{\mathcal{R}}^d, \mathbf{I}_{\mathcal{R}} \rangle$, where $s_{\text{reset}} \in \mathcal{S}_{\mathcal{R}}$, and the transition function is defined for all $s, s' \in \mathcal{S}_{\mathcal{R}}, a \in \mathcal{A}_{\mathcal{R}}$ as

$$\mathbf{P}_{\mathcal{R}}^d(s' | s, a) = \begin{cases} \mathbf{P}_{\mathcal{R}}(s' | s, a) & \text{if } s \notin \mathcal{O}_{\mathcal{R}}(d), \\ 1 & \text{if } s \in \mathcal{O}_{\mathcal{R}}(d) \text{ and } s' = s_{\text{reset}}, \text{ and} \\ 0 & \text{otherwise.} \end{cases} \quad (5.12)$$

We define $\bar{\mathbf{P}}_{\mathcal{R}}^d$ similarly for $\bar{\mathcal{M}}_{\mathcal{R}}$ when the direction d is considered.

In the following lemma, we show that the *global* transition loss $L_{\mathbf{P}}^{\pi}$ (the one associated with the high-level controller π in the MDP plan) can be bounded by some constants multiplied by the sum of the entrance loss and the expected transition losses, where we range over the rooms visited under the high-level controller. Then, minimizing the

entrance loss and the transition loss of each room guarantees that the global transition loss is minimized.

Lemma 24: Reusable transition losses

Let $\pi = \langle \omega, \Pi \rangle$ be a hierarchical controller in $\mathcal{H}$ where ω has finite memory of size $|\mathcal{V}|$, and π be the equivalent stationary policy in the MDP plan $\mathcal{M}_\Pi$. Assume that

- (i) Π only consists of latent policies; and
- (ii) for any training room $\mathcal{R} \in \text{Rooms}$ and direction $d \in \text{Dir}_{\mathcal{R}}$, the projection^a of the BSCC associated with π in $\mathcal{M}_\Pi$ to $\mathcal{S}_{\mathcal{R}}$ is included in the BSCC associated with the low-level policy $\bar{\pi}_{\mathcal{R},d}$ of $\mathcal{R}_d$.

Let

$$\begin{aligned} \mathcal{S}_{\mathcal{R},d} &= \{ \langle s, v, u \rangle \in \mathcal{S}_\Pi \mid \ell(v) = \mathcal{R} \text{ and } \langle v, u \rangle = d \}, \\ \xi_\pi(s_{\text{reset}} \mid \mathcal{R}, d) &= \mathbb{E}_{\langle s, v, u \rangle, a \sim \xi_\pi} \left[\mathbb{P}_\Pi(s_{\text{reset}} \mid \langle s, v, u \rangle, a) \mid \mathcal{S}_{\mathcal{R},d} \right], \text{ and} \\ \xi_{\text{reset}}^{\min} &= 1 - \max_{\mathcal{R} \in \text{Rooms}, d \in \text{Dir}} \left[\xi_\pi(s_{\text{reset}} \mid \mathcal{S}_{\mathcal{R},d}) + \xi_\pi(\mathcal{O}_{\mathcal{R}}(d) \times \{d\} \mid \mathcal{S}_{\mathcal{R},d}) \right]. \end{aligned}$$

Then, there is a $\kappa \geq 0$ with

$$L_{\mathbf{P}}^\pi \leq L_I + \frac{\kappa}{\xi_{\text{reset}}^{\min}} \mathbb{E}_{\mathcal{R}, d \sim \xi_\pi} L_{\mathbf{P}}^{\mathcal{R},d}.$$

Furthermore, define the expected entrance function in room $\mathcal{R}$ as

$$\mathbf{I}_{\mathcal{R}}^\pi(s) = \mathbb{E}_{\dot{s}, \langle u, v \rangle \sim \xi_\pi} \left[\mathcal{I}_{\mathcal{R}}(s \mid d = \langle u, v \rangle) \mid \dot{s} \in \mathcal{O}_{\ell(u)}(\langle u, v \rangle) \text{ and } \ell(v) = \mathcal{R} \right]$$

for any $s \in \mathcal{S}_{\mathcal{R}}$. If $\text{supp}(\mathbf{I}_{\mathcal{R}}) = \text{supp}(\mathbf{I}_{\mathcal{R}}^\pi)$, κ can be set to the maximum probability ratio of room entry during training and synthesis:

$$\kappa = \max_{\mathcal{R} \in \text{Rooms}} \left[\max_{s \in \text{supp}(\mathbf{I}_{\mathcal{R}})} \max \left\{ \frac{\mathbf{I}_{\mathcal{R}}^\pi(s)}{\mathbf{I}_{\mathcal{R}}(s)}, \frac{\mathbf{I}_{\mathcal{R}}(s)}{\mathbf{I}_{\mathcal{R}}^\pi(s)} \right\} \right]^{|S|}.$$

^aFormally speaking, we consider the projection to $\mathcal{S}_{\mathcal{R}}$ of the intersection of the BSCC of $\mathcal{M}_\Pi$ with $\mathcal{S}_{\mathcal{R}} \times \text{Dir}_{\mathcal{R}}$

Proof. For simplicity, assume that the reset state in $\mathcal{S}_\Pi$ is a triplet of the form $\langle s_{\text{reset}}, v, v_0 \rangle$ so that $\langle v, v_0 \rangle = d_0$ and $\mathcal{O}_{\ell(v_{\text{reset}})}(d_0) = \{s_{\text{reset}}\}$. We also may write $\phi(\bar{s})$ for $\phi_{\mathcal{R}}(\bar{s})$ when it is clear from the context that $\bar{s} \in \mathcal{S}_{\mathcal{R}}$. We respectively denote the marginal stationary distribution of states and directions by $\xi_\pi(s) = \mathbb{E}_{s', v, u \sim \xi_\pi} [\mathbb{1}\{s = s'\}]$

and $\xi_\pi(d) = \mathbb{E}_{s,v,u \sim \xi_\pi} [\mathbb{1}\{d = \langle v, u \rangle\}]$, respectively. Furthermore, given a direction $d \in \mathcal{E}$, we denote the conditional stationary distribution by

$$\begin{aligned} \xi_\pi(s, a | d) &= \mathbb{E}_{s', v, u \sim \xi_\pi} \left[\bar{\pi}_{\ell(v), d}(a | \phi(s)) \cdot \mathbb{1}\{s = s'\} \mid \{\langle s', v, u \rangle \in \mathcal{S}_\Pi \mid \langle v, u \rangle = d\} \right] \\ &= \mathbb{E}_{s', v, u \sim \xi_\pi} \left[\bar{\pi}_{\ell(v), d}(a | \phi(s)) \cdot \mathbb{1}\{s = s'\} \frac{\mathbb{1}\{d = \langle v, u \rangle\}}{\xi_\pi(v, u)} \right] \end{aligned}$$

In the following, we also write $\mathbf{P}(s' | s, a)$ as shorthand for $\mathbf{P}(\langle s', v \rangle | \langle s, v \rangle, a)$ (the transition function of the explicit MDP of $\mathcal{H}$) if and only if $s, s' \in \mathcal{S}_{\ell(v)}$ and $s \notin \mathcal{O}_{\ell(v)}(d)$ for some $v \in \mathcal{V}, d \in \text{out}(v)$. Denote by $\bar{\mathbf{P}}_\Pi$ the latent transition function of the *latent MDP plan* $\bar{\mathcal{M}}_\Pi$, constructed from the collection of low-level policies Π , the latent rooms $\{\bar{\mathcal{M}}_\mathcal{R} : \mathcal{R} \in \text{Rooms}\}$, and the latent entrance functions $\{\bar{\mathcal{I}}_\mathcal{R} : \mathcal{R} \in \text{Rooms}\}$. Then:

$$\begin{aligned} &L_{\mathbf{P}}^{\pi=(\omega, \Pi)} \\ &= \frac{1}{2} \mathbb{E}_{\langle s, v, u \rangle, a \sim \xi_\pi} \left\| \phi \mathbf{P}_\Pi(\cdot | \langle s, v, u \rangle, a) - \bar{\mathbf{P}}_\Pi(\cdot | \langle \phi(s), v, u \rangle, a) \right\|_1 \\ &= \frac{1}{2} \mathbb{E}_{s, v, u \sim \xi_\pi} \left[\mathbb{1}\{s \neq s_{\text{reset}}\} \mathbb{1}\{s \notin \mathcal{O}_{\ell(v)}(\langle v, u \rangle)\} \right. \\ &\quad \left. \left\| \phi \mathbf{P}_\Pi(\cdot | \langle s, v, u \rangle, *) - \bar{\mathbf{P}}_\Pi(\cdot | \langle \phi(s), v, u \rangle, *) \right\|_1 \right] \\ &\quad + \frac{1}{2} \mathbb{E}_{\langle s, v, u \rangle, d' \sim \xi_\pi} \left[\mathbb{1}\{s \neq s_{\text{reset}}\} \mathbb{1}\{s \in \mathcal{O}_{\ell(v)}(\langle v, u \rangle)\} \right. \\ &\quad \left. \left\| \phi \mathbf{P}_\Pi(\cdot | \langle s, v, u \rangle, d') - \bar{\mathbf{P}}_\Pi(\cdot | \langle \phi(s), v, u \rangle, d') \right\|_1 \right] \\ &\quad + \frac{1}{2} \mathbb{E}_{s, v, u \sim \xi_\pi} \left[\mathbb{1}\{s = s_{\text{reset}}\} \left\| \phi \mathbf{P}_\Pi(\cdot | \langle s, v, u \rangle, *) - \bar{\mathbf{P}}_\Pi(\cdot | \langle \phi(s), v, u \rangle, *) \right\|_1 \right] \quad (\pi \text{ is proper}) \\ &= \frac{1}{2} \mathbb{E}_{s, v, u \sim \xi_\pi} \left[\mathbb{1}\{s \neq s_{\text{reset}}\} \mathbb{1}\{s \notin \mathcal{O}_{\ell(v)}(\langle v, u \rangle)\} \right. \\ &\quad \left. \left\| \mathbb{E}_{a \sim \bar{\pi}_{\ell(v), (v, u)}(\cdot | \phi(s))} \left[\phi \mathbf{P}(\cdot | s, a) - \bar{\mathbf{P}}(\cdot | \phi(s), a) \right] \right\|_1 \right] \\ &\quad \quad \quad \text{(by definition of } \bar{\mathcal{M}}_\Pi \text{)} \end{aligned}$$

$$\begin{aligned}
 & + \frac{1}{2} \mathbb{E}_{\langle s, v, u \rangle, d' \sim \xi_\pi} \left[\mathbb{1} \{s \neq s_{reset}\} \mathbb{1} \{s \in \mathcal{O}_{\ell(v)}(\langle v, u \rangle)\} \right. \\
 & \qquad \qquad \qquad \left. \left\| \phi \mathcal{I}_{\ell(u)}(\cdot \mid \langle v, u \rangle) - \bar{\mathcal{I}}_{\ell(u)}(\cdot \mid \langle v, u \rangle) \right\|_1 \right] \\
 & \qquad \qquad \qquad \text{(by definition of } \mathcal{M}_\Pi \text{ ii)} \\
 & + \frac{1}{2} \mathbb{E}_{s, v, u \sim \xi_\pi} \left[\mathbb{1} \{s = s_{reset}\} \left\| \phi \mathcal{I}_{\ell(v_0)}(\cdot \mid d_0) - \bar{\mathcal{I}}_{\ell(v_0)}(\cdot \mid d_0) \right\|_1 \right] \quad \text{(by definition of } \mathcal{M}_\Pi \text{ iii)} \\
 & = \frac{1}{2} \mathbb{E}_{s, v, u \sim \xi_\pi} \left[\mathbb{1} \{s \neq s_{reset}\} \mathbb{1} \{s \notin \mathcal{O}_{\ell(v)}(\langle v, u \rangle)\} \right. \\
 & \qquad \qquad \qquad \left. \left\| \mathbb{E}_{a \sim \bar{\pi}_{\ell(v), \langle v, u \rangle}(\cdot \mid \phi(s))} \left[\phi \mathbf{P}(\cdot \mid s, a) - \bar{\mathbf{P}}(\cdot \mid \phi(s), a) \right] \right\|_1 \right] \\
 & + \frac{1}{2} \mathbb{E}_{\mathcal{R}, d \sim \xi_\pi} \left\| \phi \mathcal{I}_{\mathcal{R}}(\cdot \mid d) - \bar{\mathcal{I}}_{\mathcal{R}}(\cdot \mid d) \right\|_1 \\
 & = \frac{1}{2} \mathbb{E}_{s, v, u \sim \xi_\pi} \left[\mathbb{1} \{s \neq s_{reset}\} \mathbb{1} \{s \notin \mathcal{O}_{\ell(v)}(\langle v, u \rangle)\} \right. \\
 & \qquad \qquad \qquad \left. \left\| \mathbb{E}_{a \sim \bar{\pi}_{\ell(v), \langle v, u \rangle}(\cdot \mid \phi(s))} \left[\phi \mathbf{P}(\cdot \mid s, a) - \bar{\mathbf{P}}(\cdot \mid \phi(s), a) \right] \right\|_1 \right] \\
 & + L_I \\
 & \leq \frac{1}{2} \mathbb{E}_{s, v, u \sim \xi_\pi} \mathbb{E}_{a \sim \bar{\pi}_{\ell(v), \langle v, u \rangle}(\cdot \mid \phi(s))} \left[\mathbb{1} \{s \neq s_{reset}\} \mathbb{1} \{s \notin \mathcal{O}_{\ell(v)}(\langle v, u \rangle)\} \right. \\
 & \qquad \qquad \qquad \left. \left\| \phi \mathbf{P}(\cdot \mid s, a) - \bar{\mathbf{P}}(\cdot \mid \phi(s), a) \right\|_1 \right] + L_I \\
 & \qquad \qquad \qquad \text{(Jensen's inequality)} \\
 & = \frac{1}{2} \mathbb{E}_{d \sim \xi_\pi} \mathbb{E}_{s, a \sim \xi_\pi(\cdot \mid d)} \left[\mathbb{1} \{s \neq s_{reset}\} \mathbb{1} \{s \notin \mathcal{O}_{\ell(v)}(d)\} \right. \\
 & \qquad \qquad \qquad \left. \left\| \phi \mathbf{P}(\cdot \mid s, a) - \bar{\mathbf{P}}(\cdot \mid \phi(s), a) \right\|_1 \right] + L_I \\
 & \qquad \qquad \qquad (\star)
 \end{aligned}$$

Now, let $d = \langle v, u \rangle \in \mathcal{E}$ be a *target direction* for the room $\mathcal{R} = \ell(v)$. We consider the room $\mathcal{R}$ as an episodic MDP (cf. Assumption 6) where (i) the initial distribution corresponds to the expected entrance probabilities in $\mathcal{R}$ under π : for any $s \in \mathcal{S}_{\mathcal{R}}$

$$\mathbf{I}_{\mathcal{R}}^\pi(s) = \mathbb{E}_{\dot{s}, \langle \dot{u}, \dot{v} \rangle \sim \xi_\pi} \left[\mathcal{I}_{\mathcal{R}}(s \mid d_I = \langle \dot{u}, \dot{v} \rangle) \mid \dot{s} \in \mathcal{O}_{\ell(\dot{u})}(\langle \dot{u}, \dot{v} \rangle) \text{ and } \dot{v} = v \right]$$

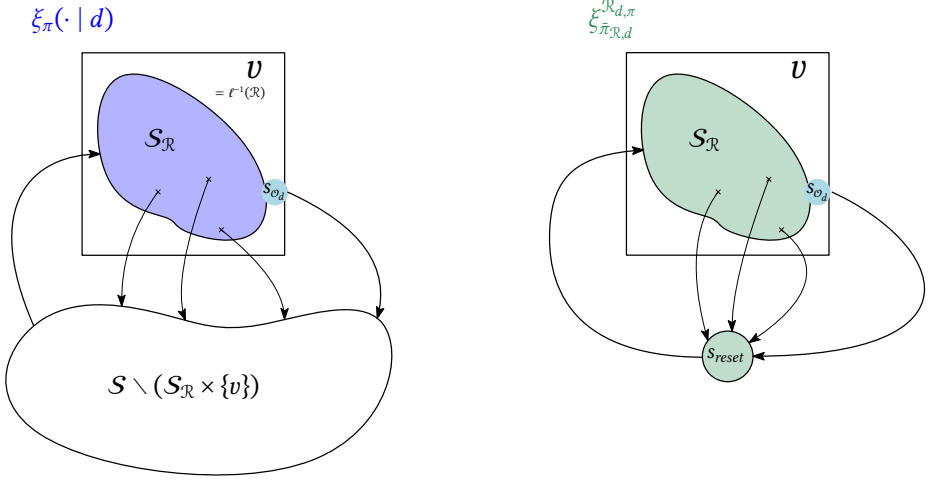


Figure 5.6: Room $\mathcal{R} = \ell(v)$ in the hierarchical model (left) and the same room taken individually (right). Both distributions $\xi_{\pi}(\cdot | d)$ and $\xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}$ correspond to the limiting distributions over $\mathcal{S}_{\mathcal{R}}$ when $\bar{\pi}_{\mathcal{R},d}$ is executed in $\mathcal{R}$. The sole difference remains in the fact that the reset is considered outside $\mathcal{R}$ in the hierarchical model (Assumption 12) while it is considered to be part of the state space when $\mathcal{R}$ is taken individually (Assumption 6).

(where d_l is the direction from which $\mathcal{R}$ is entered); and (ii) the room is reset when an exit state in direction d is visited: for any $s, s' \in \mathcal{S}_{\mathcal{R}}$, $a \in \mathcal{A}_{\mathcal{R}}$,

$$\mathbf{P}_{\mathcal{R}}^{d,\pi}(s' | s, a) = \begin{cases} 1 & \text{if } s' = s_{reset} \text{ and } s \in \mathcal{O}_{\mathcal{R}}(d), \\ \mathbf{I}_{\mathcal{R}}^{\pi}(s') & \text{if } s = s_{reset}, \text{ and} \\ \mathbf{P}_{\mathcal{R}}(s' | s, a) & \text{otherwise.} \end{cases} \quad (5.13)$$

We call the resulting MDP the *individual room* version of $\mathcal{R}$ that we denote by $\mathcal{R}_{d,\pi}$. The stationary distribution of the room $\mathcal{R}_{d,\pi}$ for the low-level policy $\bar{\pi}_{\mathcal{R},d}$ is $\xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}$. Observe that $\xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}$ is over $\mathcal{S}_{\mathcal{R}}$, which includes the reset state s_{reset} , while $\xi_{\pi}(\cdot | d)$ is over the exact same state space but without the reset state (since the reset state is a special state outside $\mathcal{R}$, shared by all the rooms in the hierarchical model; cf. Assumption 12 and the definition of $\mathcal{M}_{\Pi}$).

Furthermore, notice that, modulo this reset state, the two distributions are the same (see Figure 5.6): they both consist of the limiting distribution over $\mathcal{S}_{\mathcal{R}}$ when $\bar{\pi}_{\mathcal{R},d}$ is

executed in $\mathcal{R}$. All the transition distributions remain the same, except those of the exit states: in the hierarchical model $\mathcal{H}$, every state $s \in \mathcal{O}_{\mathcal{R}}(d = \langle v, u \rangle)$ transitions to u deterministically, while in the individual room $\mathcal{R}_{d,\pi}$, they transition to the reset state deterministically. Still, in both cases, $\mathcal{R}$ is entered and exited with the same probability (respectively from and to $(S \setminus S_{\mathcal{R}} \times \{v\})$ in $\mathcal{H}$ and s_{reset} in the individual room $\mathcal{R}_{d,\pi}$). Therefore, we have:

$$\xi_{\pi}(s \mid d) = \xi_{\pi_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}(s \mid S_{\mathcal{R}} \setminus \{s_{reset}\}) = \frac{\xi_{\pi_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}(s) \cdot \mathbb{1}\{s \neq s_{reset}\}}{1 - \xi_{\pi_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}(s_{reset})}. \quad (5.14)$$

Instead of sampling from $\xi_{\pi}(s \mid d)$ in Equation $\star$, we would rather like to sample from the distribution of the individual room $\xi_{\pi_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}(s \mid S_{\mathcal{R}} \setminus \{s_{reset}\})$. We have:

$$\begin{aligned} & \mathbb{E}_{s,a \sim \xi_{\pi}(\cdot \mid d)} \left[\mathbb{1}\{s \neq s_{reset}\} \mathbb{1}\{s \notin \mathcal{O}_{\ell(v)}(d)\} \|\phi \mathbf{P}(\cdot \mid s, a) - \bar{\mathbf{P}}(\cdot \mid \phi(s), a)\|_1 \right] \\ &= \sum_{s \in S} \sum_{a \in \mathcal{A}} \left[\xi_{\pi}(s \mid d) \bar{\pi}_{\mathcal{R},d}(a \mid \phi(s)) \right. \\ & \quad \left. \mathbb{1}\{s \neq s_{reset}\} \mathbb{1}\{s \notin \mathcal{O}_{\ell(v)}(d)\} \|\phi \mathbf{P}(\cdot \mid s, a) - \bar{\mathbf{P}}(\cdot \mid \phi(s), a)\|_1 \right] \end{aligned} \quad (5.15)$$

$$\begin{aligned} &= \sum_{s \in S} \sum_{a \in \mathcal{A}} \left[\frac{\xi_{\pi_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}(s) \mathbb{1}\{s \neq s_{reset}\}}{1 - \xi_{\pi_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}(s_{reset})} \bar{\pi}_{\mathcal{R},d}(a \mid \phi(s)) \right. \\ & \quad \left. \mathbb{1}\{s \neq s_{reset}\} \mathbb{1}\{s \notin \mathcal{O}_{\ell(v)}(d)\} \|\phi \mathbf{P}_{\mathcal{R}}(\cdot \mid s, a) - \bar{\mathbf{P}}_{\mathcal{R}}(\cdot \mid \phi(s), a)\|_1 \right] \end{aligned} \quad (5.16)$$

Notice that we can pass from Equation 5.15 to Equation 5.16 because we only consider states $s \neq s_{reset}$ and $s \notin \mathcal{O}_{\ell(v)}(d)$. States that do not satisfy both constraints are the only ones for which $\mathbf{P}(\cdot \mid s, a)$ differs from $\mathbf{P}_{\mathcal{R}}^{d,\pi}(\cdot \mid s, a)$ (Equation 5.13). Furthermore, in that case, we have $\mathbf{P}_{\mathcal{R}}^{d,\pi}(\cdot \mid s, a) = \mathbf{P}_{\mathcal{R}}(\cdot \mid s, a)$. Then we have:

$$\begin{aligned} & \mathbb{E}_{s,a \sim \xi_{\pi_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}} \left[\frac{\mathbb{1}\{s \neq s_{reset}\}}{1 - \xi_{\pi_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}(s_{reset})} \mathbb{1}\{s \notin \mathcal{O}_{\ell(v)}(d)\} \|\phi \mathbf{P}_{\mathcal{R}}(\cdot \mid s, a) - \bar{\mathbf{P}}_{\mathcal{R}}(\cdot \mid \phi(s), a)\|_1 \right] \\ &= \mathbb{E}_{s,a \sim \xi_{\pi_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}} \left[\frac{\mathbb{1}\{s \neq s_{reset}\}}{1 - \xi_{\pi_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}(s_{reset})} \|\phi \mathbf{P}_{\mathcal{R}}^d(\cdot \mid s, a) - \bar{\mathbf{P}}_{\mathcal{R}}^d(\cdot \mid \phi(s), a)\|_1 \right] \\ & \quad \text{(by definition of } \mathbf{P}_{\mathcal{R}}^d \text{ and } \bar{\mathbf{P}}_{\mathcal{R}}^d, \text{ cf. Equation 5.12)} \end{aligned}$$

$$\begin{aligned}
 &= \frac{1}{1 - \xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}(s_{reset})} \mathbb{E}_{s,a \sim \xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}} \left[\mathbb{1}\{s \neq s_{reset}\} \left\| \phi \mathbf{P}_{\mathcal{R}}^d(\cdot | s, a) - \bar{\mathbf{P}}_{\mathcal{R}}^d(\cdot | \phi(s), a) \right\|_1 \right] \\
 &\leq \frac{1}{1 - \xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}(s_{reset})} \mathbb{E}_{s,a \sim \xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}} \left\| \phi \mathbf{P}_{\mathcal{R}}^d(\cdot | s, a) - \bar{\mathbf{P}}_{\mathcal{R}}^d(\cdot | \phi(s), a) \right\|_1
 \end{aligned}$$

Assuming that the projection of the BSCC associated with π in $\mathcal{M}_{\Pi}$ to $\mathcal{S}_{\mathcal{R}}$ is included into the BSCC associated with the latent policy $\bar{\pi}_{\mathcal{R},d}$ in room $\mathcal{R}$, we have that $\text{supp}(\xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}) \subseteq \text{supp}(\xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}})$, where $\xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}}$ denotes the stationary distribution of the training room $\mathcal{R}_d$ under the latent policy $\bar{\pi}_{\mathcal{R},d}$. Then:

$$\begin{aligned}
 &\frac{1}{1 - \xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}(s_{reset})} \mathbb{E}_{s,a \sim \xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}} \left\| \phi \mathbf{P}_{\mathcal{R}}^d(\cdot | s, a) - \bar{\mathbf{P}}_{\mathcal{R}}^d(\cdot | \phi(s), a) \right\|_1 \\
 &= \frac{1}{1 - \xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}(s_{reset})} \sum_{s \in \text{supp}(\xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}})} \sum_{a \in \mathcal{A}_{\mathcal{R}}} \left[\xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}(s) \bar{\pi}_{\mathcal{R},d}(a | \phi(s)) \right. \\
 &\quad \left. \left\| \phi \mathbf{P}_{\mathcal{R}}^d(\cdot | s, a) - \bar{\mathbf{P}}_{\mathcal{R}}^d(\cdot | \phi(s), a) \right\|_1 \right] \\
 &= \frac{1}{1 - \xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}(s_{reset})} \sum_{s \in \text{supp}(\xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}})} \sum_{a \in \mathcal{A}_{\mathcal{R}}} \left[\frac{\xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}(s)}{\xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}}(s)} \xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}}(s) \bar{\pi}_{\mathcal{R},d}(a | \phi(s)) \right. \\
 &\quad \left. \left\| \phi \mathbf{P}_{\mathcal{R}}^d(\cdot | s, a) - \bar{\mathbf{P}}_{\mathcal{R}}^d(\cdot | \phi(s), a) \right\|_1 \right] \\
 &= \frac{1}{1 - \xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}(s_{reset})} \mathbb{E}_{s,a \sim \xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}} \left[\frac{\xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}(s)}{\xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}}(s)} \left\| \phi \mathbf{P}_{\mathcal{R}}^d(\cdot | s, a) - \bar{\mathbf{P}}_{\mathcal{R}}^d(\cdot | \phi(s), a) \right\|_1 \right] \\
 &\leq \frac{1}{1 - \xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}(s_{reset})} \mathbb{E}_{s,a \sim \xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}} \left[\max_{s' \in \text{supp}(\xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}})} \left[\frac{\xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}(s')}{\xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}}(s')} \right] \left\| \phi \mathbf{P}_{\mathcal{R}}^d(\cdot | s, a) - \bar{\mathbf{P}}_{\mathcal{R}}^d(\cdot | \phi(s), a) \right\|_1 \right] \\
 &= \frac{1}{1 - \xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}(s_{reset})} \max_{s \in \text{supp}(\xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}})} \left[\frac{\xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}(s)}{\xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}}(s)} \right] \mathbb{E}_{s,a \sim \xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}} \left\| \phi \mathbf{P}_{\mathcal{R}}^d(\cdot | s, a) - \bar{\mathbf{P}}_{\mathcal{R}}^d(\cdot | \phi(s), a) \right\|_1 \\
 &= \max_{s \in \text{supp}(\xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}})} \left[\frac{\xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}(s)}{\xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}}(s)} \right] \frac{2L_{\mathbf{P}}^{\mathcal{R},d}}{1 - \xi_{\bar{\pi}_{\mathcal{R},d}}^{\mathcal{R}_{d,\pi}}(s_{reset})}
 \end{aligned}$$

If the initial distributions of the individual room $\mathcal{R}_{d,\pi}$ and the training room $\mathcal{R}_d$ have the same support, then the projection and the BSCCs coincide since the same set of states is eventually visited under $\bar{\pi}$ from states of $\text{supp}(\mathbf{I}_{\mathcal{R}}) = \text{supp}(\mathbf{I}_{\mathcal{R}}^{\mathcal{I}})$. Furthermore,

by [O’Cinneide, 1993, Theorem 1], we have

$$\max_{s \in \text{supp}(\xi_{\pi, \mathcal{R}, d}^{\mathcal{R}})} \left[\frac{\xi_{\pi, \mathcal{R}, d}^{\mathcal{R}, \pi}(s)}{\xi_{\pi, \mathcal{R}, d}^{\mathcal{R}}(s)} \right] \quad (5.17)$$

$$\begin{aligned} &\leq \max_{s \in \text{supp}(\xi_{\pi, \mathcal{R}, d}^{\mathcal{R}})} \max \left\{ \frac{\xi_{\pi, \mathcal{R}, d}^{\mathcal{R}, \pi}(s)}{\xi_{\pi, \mathcal{R}, d}^{\mathcal{R}}(s)}, \frac{\xi_{\pi, \mathcal{R}, d}^{\mathcal{R}}(s)}{\xi_{\pi, \mathcal{R}, d}^{\mathcal{R}, \pi}(s)} \right\} \\ &\leq \left(\max_{s \in \text{supp}(\mathbf{I}_{\mathcal{R}})} \max \left\{ \frac{\mathbf{I}_{\mathcal{R}}^{\pi}(s)}{\mathbf{I}_{\mathcal{R}}(s)}, \frac{\mathbf{I}_{\mathcal{R}}(s)}{\mathbf{I}_{\mathcal{R}}^{\pi}(s)} \right\} \right)^{|\mathcal{S}|} \quad (\text{cf. Equation 5.13}) \\ &= \kappa_{\mathcal{R}, d}. \end{aligned} \quad (5.18)$$

Otherwise, we set $\kappa_{\mathcal{R}, d}$ to $\max_{s \in \text{supp}(\xi_{\pi, \mathcal{R}, d}^{\mathcal{R}})} \left[\frac{\xi_{\pi, \mathcal{R}, d}^{\mathcal{R}, \pi}(s)}{\xi_{\pi, \mathcal{R}, d}^{\mathcal{R}}(s)} \right]$. Moreover, let

$$\mathcal{S}_{\mathcal{R}, d} = \{ \langle s, v, u \rangle \in \mathcal{S}_{\Pi} \mid \ell(v) = \mathcal{R} \text{ and } \langle v, u \rangle = d \}$$

and define

$$\xi_{\pi}(s_{\text{reset}} \mid \mathcal{R}, d) = \mathbb{E}_{\langle s, v, u \rangle, a \sim \xi_{\pi}} \left[\mathbf{P}_{\Pi}(s_{\text{reset}} \mid \langle s, v, u \rangle, a) \mid \mathcal{S}_{\mathcal{R}, d} \right].$$

Notice that

$$\begin{aligned} \xi_{\pi, \mathcal{R}, d}^{\mathcal{R}, \pi}(s_{\text{reset}}) &= \mathbb{E}_{\langle s, v, u \rangle, a \sim \xi_{\pi}} \left[\mathbf{P}_{\Pi}(s_{\text{reset}} \mid \langle s, v, u \rangle, a) + \mathbb{1}\{s \in \mathcal{O}_{\mathcal{R}}(d)\} \mid \mathcal{S}_{\mathcal{R}, d} \right] \\ &= \xi_{\pi}(s_{\text{reset}} \mid \mathcal{R}, d) + \xi_{\pi}(\mathcal{O}_{\mathcal{R}}(d) \times \{d\} \mid \mathcal{S}_{\mathcal{R}, d}) \end{aligned}$$

by (i) the stationary property, (ii) definition of $\mathbf{P}_{\mathcal{R}}^{d, \pi}$ (cf. Equation 5.13 and Figure 5.6), (iii) the fact that the probability of exiting the room is equal to the probability of visiting an exit state, and (iv) the fact that resetting the room and visiting an exit state are disjoint events (when an exit state is visited, it always transitions to the next room, never to the reset state).

By putting all together, we have

$$\begin{aligned} &L_{\mathbf{P}}^{\pi} \\ &\leq L_I + \frac{1}{2} \mathbb{E}_{d \sim \xi_{\pi}} \mathbb{E}_{s, a \sim \xi_{\pi}(\cdot \mid d)} \left[\mathbb{1}\{s \neq s_{\text{reset}}\} \mathbb{1}\{s \notin \mathcal{O}_{\ell(v)}(d)\} \|\phi \mathbf{P}(\cdot \mid s, a) - \bar{\mathbf{P}}(\cdot \mid \phi(s), a)\|_1 \right] \\ &\leq L_I + \mathbb{E}_{\mathcal{R}, d \sim \xi_{\pi}} \left[\frac{\kappa_{\mathcal{R}, d} \cdot L_{\mathbf{P}}^{\mathcal{R}, d}}{1 - \xi_{\pi, \mathcal{R}, d}^{\mathcal{R}, \pi}(s_{\text{reset}})} \right] \\ &= L_I + \mathbb{E}_{\mathcal{R}, d \sim \xi_{\pi}} \left[\frac{\kappa_{\mathcal{R}, d} \cdot L_{\mathbf{P}}^{\mathcal{R}, d}}{1 - \xi_{\pi}(s_{\text{reset}} \mid \mathcal{R}, d) - \xi_{\pi}(\mathcal{O}_{\mathcal{R}}(d) \times \{d\} \mid \mathcal{S}_{\mathcal{R}, d})} \right] \end{aligned}$$

$$\begin{aligned}
&\leq L_I + \mathbb{E}_{\mathcal{R}, d \sim \xi_\pi} \left[\frac{\max \{ \kappa_{\mathcal{R}^*, d^*} : \mathcal{R}^* \in \text{Rooms}, d^* \in \text{Dir}_{\mathcal{R}^*} \} L_{\mathbf{P}}^{\mathcal{R}, d}}{1 - \max_{\mathcal{R}^* \in \text{Rooms}, d^* \in \text{Dir}_{\mathcal{R}}} [\xi_\pi(s_{\text{reset}} | \mathcal{R}^*, d^*) + \xi_\pi(\mathcal{O}_{\mathcal{R}^*}(d^*) \times \{d^*\} | \mathcal{S}_{\mathcal{R}^*, d^*})]} \right] \\
&\leq L_I^\Pi + \frac{\kappa}{\xi_{\text{reset}}^{\min}} \mathbb{E}_{\mathcal{R}, d \sim \xi_\pi} L_{\mathbf{P}}^{\mathcal{R}, d}
\end{aligned}$$

where $\kappa = \max \{ \kappa_{\mathcal{R}^*, d^*} : \mathcal{R}^* \in \text{Rooms}, d^* \in \text{Dir}_{\mathcal{R}^*} \}$ and

$$\xi_{\text{reset}}^{\min} = 1 - \max_{\mathcal{R} \in \text{Rooms}, d \in \text{Dir}_{\mathcal{R}}} [\xi_\pi(s_{\text{reset}} | \mathcal{R}, d) + \xi_\pi(\mathcal{O}_{\mathcal{R}}(d) \times \{d\} | \mathcal{S}_{\mathcal{R}, d})]. \quad (5.19)$$

This concludes the proof. $\square$

Interestingly, note that if this initial distribution is powerful enough to provide an *exact coverage* (same support) of the entrance states visited under the high-level controller π , the constant κ can be determined solely based on the ratio of the initial distributions obtained during training and synthesis.

As a consequence of Lemma 24, by additionally taking into account Equation 5.2 (the value difference bound), and Theorem 33 (the equality of the value function in the compact plan), we derive the following corollary, which coincides with the result presented in Proposition 2.

Corollary 7: Initial value difference bound for hierarchical control

Under the same assumptions as in Lemma 24,

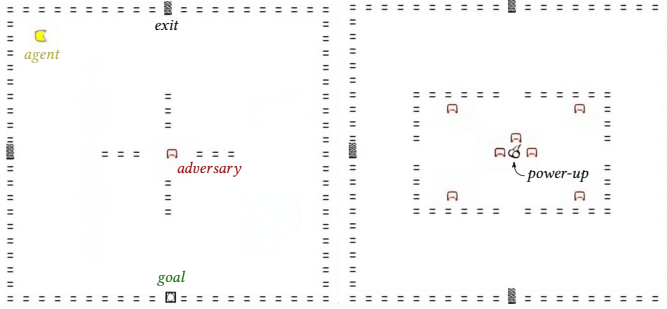
$$\left| V_{\mathcal{M}}^\pi(\varphi) - \bar{V}_I^\omega(\diamond^y \text{goal}) \right| \leq \frac{L_I + \kappa / \xi_{\text{reset}}^{\min} \cdot \mathbb{E}_{\mathcal{R}, d \sim \xi_\pi} L_{\mathbf{P}}^{\mathcal{R}, d}}{\xi_\pi(s_{\text{reset}})(1-\gamma)}.$$

4 Experimental Evaluation

Environment. In this section, we report on results from a case study. Further details on our setting is provided in Appendix D.

We use a parameterized grid-world environment with N rooms subdivided into $m \times n$ cells (cf. Figure 5.7). Cells may contain *items* such as walls, entry and exit points, power-ups, and adversaries. There are adversaries patrolling with varying behaviors between rooms, e.g., randomly or along walls; some chase the agent when entering their patrol zone. The agent has P life points, decrementing upon adversary contact or timeout. By collecting power-ups (appearing randomly with low probability), the agent

³Demonstration with 9 rooms: <https://youtu.be/crowN8-GaRg>


 Figure 5.7: Two rooms of 20×20 cells (9 rooms in Appendix D, Figure 1).³

shortly becomes invulnerable, causing adversaries to vanish upon contact. The high-level objective is to reach a target position while keeping the life points strictly positive. Episodes start from an initial position and conclude when the agent reaches the target or runs out of life points. The state space comprises two components:

- (i) a set of N 3-dimensional bitmaps $\mathbf{M} \in \{0, 1\}^{L \times m \times n}$, where each layer $l \in L$ corresponds to an item type on the grid. Entry $\mathbf{M}_{l,i,j}^R$ is 1 if and only if room R has item l in cell (i, j) ;
- (ii) step, power-up, and life-point counters $\langle a, b, c \rangle$, with $1 \leq a \leq T_1$, $1 \leq b \leq T_2$, $1 \leq c \leq P$.

DRL baseline. Clearly, the resulting state space, although finite, does not scale to traditional synthesis approaches (the state space is simply intractable). We use *convolutional neural networks* (CNNs, LeCun et al., 1989) to deal with input matrices $\mathbf{M}$ and a sparse reward signal $\mathcal{R}(s, a) = r^* \cdot (\mathbb{1}\{\ell(s) \cap \mathbf{T} \neq \emptyset\} - \mathbb{1}\{\ell(s) \cap \mathbf{B} \neq \emptyset\})$, where $r^* > 0$ is an arbitrary reward (or conversely, a penalty) obtained upon reaching the target labeled with $\mathbf{T}$ (or an undesirable state labeled with $\mathbf{B}$). To guide the agent, we add simple *potential-based reward shaping* scheme [Ng et al., 1999; Wiewiora, 2003] based on the L_1 distance to the target. The resulting potential is $\Phi(s, a, s') = \gamma\Phi(s') - \Phi(s) + \mathcal{R}(s, a)$ leading to the reward signal $\mathcal{R}_\Phi(s, a) = \mathbb{E}_{s' \sim \mathbf{P}(\cdot|s,a)} \Phi(s, a, s')$, where

$$\Phi(s) = 1 - \frac{\min \left\{ |x(t_1) - x(s)| + |y(t_2) - y(s)| : t_1, t_2 \in \ell^{-1}(\mathbf{T}) \right\}}{N \cdot (m + n)},$$

and $x(s)$, $y(s)$ respectively return the Euclidean coordinates along the horizontal and vertical axes corresponding to state $s \in \mathcal{S}$. Intuitively, $|\Phi(s) - 1|$ reflects the normalized distance of state s to the targets labeled with $\mathbf{T}$. When the agent gets closer (resp. further)

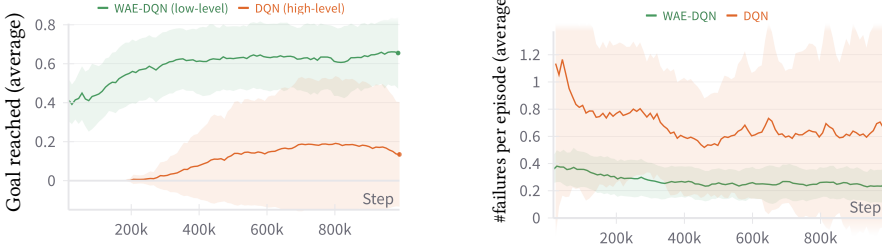


Figure 5.8: Evaluation of the policies along training steps of WAE-DQN and (vanilla) DQN in a grid world with 9 rooms of 20×20 cells. Each point is the average across 30 policy rollouts. We train five different instances of the algorithm per policy with different random seeds, where the solid line is the mean and the shaded interval is the standard deviation. To train the DQN agent, we set a time limit five times longer than that used for training rooms in the WAE-DQN agents. Furthermore, the DQN agent is equipped with 3 life points, while the WAE-DQN agents are limited to one.

to target states when executing an action, the resulting reward is positive (resp. negative). Nevertheless, as demonstrated in Figure 5.8, while DQN allows reducing contact with adversaries, the hierarchical nature of the decisions required to reach a target hinders learning the high-level objective.

Learning the low-level policies. Instead of learning a policy for the grid world directly, we run our WAE-DQN algorithm to learn the set of low-level policies Π along with their linked latent space models. Recall the representation quality guarantees of our algorithm (cf. Equation 5.3): the same latent space can be used for rooms sharing similar features. For instance, in an environment composed of 9 rooms similar to those of Figure 5.7, where the agent may exit via directions in $\{\leftarrow, \rightarrow, \uparrow, \downarrow\}$, we only train 4 latent policies (one per direction) instead of $9 \cdot 4 = 36$. For training in a room $\mathcal{R}$, we let $\mathbf{I}_{\mathcal{R}}$ uniformly distribute the agent’s possible entry positions. Note that the entrance function $\mathcal{I}_{\mathcal{R}}$ is unknown at training time. The adversaries’ initial positions are randomly distributed by $\mathbf{I}_{\mathcal{R}}$, whereas they may be different following the (unknown) function $\mathcal{I}_{\mathcal{R}}$ in the high-level model. The low-level objectives $\phi_{\mathcal{R}}^d$ specify reaching the target exit while avoiding the adversaries, before that the environment reset is triggered.

We combined WAE-MDPs with *Rainbow DQN* [Hessel et al., 2018], embedding state-of-the-art extensions and improvements of DQN. For the experiments presented in this section, we found that using a categorical network for learning a value distribution [Bellemare et al., 2023] and prioritized experience replay [Schaul et al., 2016] did not significantly improve the performance of the WAE-DQN agent.

Table 5.2: Values of the controller $\pi = \langle \bar{\omega}, \Pi \rangle$ for various sizes of the grid and $\gamma = 0.99$.

Grid Size	Life Points	# Adversaries	$\hat{V}_1(\varphi^{\gamma=1})$		$\bar{V}_1^{\bar{\omega}}(\varphi)$	$\hat{V}_1(\varphi)$		
3×3	1	11	0.5467	± 0.1017	0.1378	0.07506	± 0.01664	
3×3	3	11	0.7	± 0.09428	0.4343	0.01	± 0.00163	
5×5	3	23	0.4933	± 0.09832	0.1763	0.007833	± 0.002131	
5×5	5	23	0.5667	± 0.07817	0.346	0.00832	± 0.00288	
7×7	7	47	0.02667	± 0.01491	0.004229	$5.565 \cdot 10^{-6}$	$\pm 7 \cdot 10^{-6}$	

PAC bounds. For every latent policy $\bar{\pi}_d \in \Pi$, Table 5.1 shows the PAC bound on the transition loss $\hat{L}_P$ (cf. Lemma 14) for $\delta = 0.05$ and $\epsilon = 0.01$. Intuitively, those values serve as a measure of proximity between the original and abstracted rooms when the agent operates under latent policy $\bar{\pi}_d$ designed for target direction d . A value closer to zero indicates that the resulting behaviors of the agent in the corresponding abstracted model better resemble those in the original training room.

Table 5.1: Local loss.

Target dir. d	$\hat{L}_P$
RIGHT $\rightarrow$	0.504124
LEFT $\leftarrow$	0.777875
UP $\uparrow$	0.496315
DOWN $\downarrow$	0.480581

Synthesis. To estimate the latent entrance function, we explore the high-level environment through random execution of the low-level latent policies. We further consider MADEs (Definition 41) which are simple feed-forward networks with masked connections, allowing to learn complex distributions from a dataset. Utilizing the data collected via this exploration, we train a single MADE to learn $\bar{\mathcal{I}}_{\mathcal{R}}$ for all rooms $\mathcal{R} \in \text{Rooms}$. We finally construct $\bar{\mathcal{M}}_{\Pi}^{\mathcal{G}}$ (cf. Section 3.1) and apply the synthesis procedure to obtain a hierarchical controller $\pi = \langle \bar{\omega}, \Pi \rangle$.

Table 5.2 reports the values of π obtained for various environment sizes. Respectively, $\hat{V}_1(\varphi^{\gamma=1})$, $\bar{V}_1^{\bar{\omega}}$, and $\hat{V}_1(\varphi)$ refer to the values of the objective: (i) approximated and *undiscounted* in the original model, (ii) computed via value iteration in $\bar{\mathcal{M}}_{\Pi}^{\mathcal{G}}$, and (iii) approximated in the original model. Approximated values are averaged over 30 rollouts for different random seeds. The latent policies are *solely* trained in a grid of 9 rooms with 11 adversaries, while they are used for synthesizing controllers in larger environments, respectively with 25, 49 rooms and 23, 47 adversaries. This highlights the reusability of the low-level policies in scenarios of larger scales when the rooms share similar features as those used during training.

5 Concluding Discussion

Our approach integrates DRL with reactive synthesis to obtain a hierarchical controller for reach-avoid specifications. First, we train low-level policies and an abstract model

via WAE-DQN. Then, we compute the value of the specification for the high-level MDP, optionally estimating PAC bounds. Finally, we synthesize a high-level controller in the resulting high-level MDP, preserving the bounds. Experiments show the feasibility of our approach in hierarchical scenarios which are hard to handle by pure DRL.

Taking inspiration from goal-conditioned RL (e.g., Andrychowicz et al., 2017) and ensemble modeling [Janner et al., 2019; Lu et al., 2022; Alegre et al., 2023] would enable unifying the learning of the latent policies, further allowing learning of the latent entrance function in the same procedure (“end-to-end”).

Conclusion

This dissertation was motivated by two key observations. Firstly, deep reinforcement learning (deep RL) is incredibly powerful in providing intelligent agents with control policies in complex, high-dimensional environments. However, they lack guarantees, which is a major issue hampering their widespread deployment in safety-critical scenarios. Secondly, model checking is a very useful technique for providing formal guarantees on system behavior, but it struggles to scale to high-dimensional state and action spaces, which limits its applicability to complex scenarios. The combination of the strengths of deep RL and model checking allows for bypassing the limitations of both approaches while retaining their mutual benefits: providing formal guarantees on the agent's behaviors induced by deep RL policies in complex environments.

The question was precisely how to combine them. In this dissertation, we addressed this question and *we further enabled formal verification of deep RL policies*.

To allow for the activation of formal verification, we developed abstraction techniques to learn latent models that are amenable to model checking. We introduced two generative models that learn a latent space representation of the environment: Variational and Wasserstein auto-encoded Markov decision processes (VAE- and WAE-MDPs). While the former allows learning a verifiable latent abstraction, the latter improves on robustness and stability of the training procedure, backed up with additional theoretical learning guarantees. Furthermore, we showed that both approaches yield a latent model that is bisimilarly close to the original environment when its training

loss function is minimized. The bisimulation distance is a (pseudo-) metric that allows quantifying the behavioral difference between models and is our key ingredient to lift the guarantees resulting from model checking the latent model to the original environment. We further provided probably approximately correct (PAC) estimation schemes to evaluate the closeness of the latent model to the original environment in terms of bisimulation.

Beyond distillation, we showed that WAE-MDPs are also theoretically grounded for providing online learning guarantees, which opened possibilities to use the framework as a general latent space learner for deep RL. Notably, we demonstrated that our method is theoretically grounded for learning representations in deep RL, facilitating policy learning in complex environments. In particular, we considered partially observable environments, where representation learning is crucial. In partially observable systems, the agent cannot directly observe the environment’s state but receives imperfect, non-Markovian observations. To make informed decisions, the agent must either remember the entire sequence of observations and actions (the “history”) or maintain a belief distribution over the possible unobservable states of the system. However, maintaining this belief is often intractable, while the history-based approach becomes quickly impractical as the history space grows exponentially with the time horizon.

Most deep RL approaches rely on recurrent neural networks to compress the history into a fixed-size representation, but this lacks representation guarantees. In contrast, our approach learns to maintain a belief in a principled way, supported by a WAE-MDP learned in parallel to the policy. We demonstrated that combining WAE-MDPs with policy learning enables generating a compact, informative representation of the policy inputs, which is guaranteed to facilitate its optimization.

Finally, we addressed the problem of synthesizing high-level controllers for very large environments, where a high-level description of the latter is available. Synthesis is an automated technique for generating controllers that satisfy a given specification by design. Given a high-level description of the environment and collections of low-level policies, latent models, as well as reach-avoid specifications, we showed how to synthesize a hierarchical controller whose decisions consist of the choices of low-level policies to execute, by composing the latent models. This composition poses several challenges. First, we showed that the problem becomes non-Markovian due to the hierarchical nature of the environment. Second, even if the synthesis is performed in the unified latent space of the models, the resulting state space remains too large to be explored entirely. Third, stitching the latent models together may lead to a loss of the bisimulation guarantees equipped with each individual model. We addressed these challenges by demonstrating that the synthesis problem reduces to constructing a controller in a compact, tractable model based on the high-level description of the environment and a small memory. Furthermore, we proved that the guarantees of the latent models are reusable under some mild assumptions, allowing lifting them to the

unified model and finally enabling synthesis within very large environments with reach-avoid specifications.

Perspectives

This thesis mainly focused on enabling formal verification of deep RL policies. The activation of formal verification now opens diverse perspectives for future research.

Emulating Synthesis through Learning

As mentioned several times throughout this manuscript, the applicability of synthesis is limited by the complexity of the system, the size of the specification, and the requirement of an explicit model of the environment. Trading traditional (intractable) synthesis procedures for deep RL approaches with guarantees “end-to-end”, i.e., in a unique dedicated and automated learning instance, has diverse avenues for future research.

The aforementioned tractability issues have been addressed in the context of model checking, in the framework presented in Chapter 3, which allows formally verifying deep RL policies that have been learned *a priori*. While this is a significant step forward, the method “only” tells us the *degree of satisfaction* of the specification (through values and PAC guarantees) for a given, pre-learned policy. Notably, a violation of the specification may indicate that the policy is unsafe and far from optimal, that it is not robust to the environment dynamics, or that the reward signal designed by the user is not appropriate to learn the specification. In case of failure, the framework does not yet provide any *automated* mechanism for refining the learning procedure to eventually lead to the satisfaction of the specification. Such mechanisms would be a significant step towards the automated generation of safe and correct policies through learning. Possible directions include local corrections of the policy, in the same spirit as the work of Pathak et al. [2015]. The challenge will be to find a way to correct the policy without causing the loss of the (bisimulation) guarantees provided by the latent model — otherwise, the result of the correction would turn obsolete and disconnected from the original environment, requiring a new epoch of latent model learning.

In Chapter 5, we discussed the application of synthesis for reach-avoid specifications within large MDPs, where a high-level description of the environment is available. The approach involves learning a collection of low-level policies and latent models that are subsequently combined by the synthesis method to meet the specification. Although the method allows learning the collection of policies in parallel, it is not “end-to-end”, in the sense that the synthesis step only occurs when the different components have been learned and fixed. Thus, it is not fully automated.

Taking a step back, *emulating synthesis through learning*, i.e., automating the learning of a specification within an unknown, intractable environment *with verifiable guarantees*, builds on the following requirements:

1. one needs to make sure that the input reward signal is appropriate to learn the specification;
2. desirably, the guarantees should hold *online*, i.e., one should be able to track the degree of satisfaction of the input specification during the learning process;
3. the process should guarantee the satisfaction of this specification, i.e., that the policy spit out by the learning procedure meets the input requirements; and
4. all three previous steps should be automated.

Requirement 1 is already intricate, as it requires a deep understanding of the environment and the specification to design a reward signal that is appropriate for learning the desired behavior. The automated design of such reward signals is an ongoing research topic (e.g., Sadigh et al., 2014; Camacho et al., 2019; Jothimurugan et al., 2021a; Alur et al., 2023; Shao and Kwiatkowska, 2023; Elbarbari et al., 2022). However, none of these works generate a reward signal that is fully compliant with VLC formulae, most of them focus on LTL (not learnable, see Chapter 1, Section 3.4) or LTL_f (see Chapter 1, Section 3.5 for a brief discussion). Ensuring the automated generation of reward signals that are appropriate for learning VLC formulae is a promising direction since VLC is learnable and has a direct link with bisimulation (Theorem 21), which is compliant with our latent model framework.

While requirement 2 is a challenging task, as discussed in Chapter 3, Section 3, our WAE-MDP framework unlocks the possibility of learning the latent model *online*, i.e., at the same time as the (latent) policy (Theorem 28). Then, tracking the satisfaction of the specification is enabled. Being provided with the latent model during learning further unlocks the possibility of tackling *safe RL* from a formal methods standpoint (e.g., via probabilistic shields, Alshiekh et al., 2018; Jansen et al., 2020).

Requirement 3 can be enabled with future research on the refinement of the learning process, as discussed above.

The Latent MDP is a World Model

We already mentioned it, WAE-MDPs can be learned online, at the same time as the policy. The related latent model comes with abstraction and representation guarantees. A feature that we did not explicitly use in this thesis but that is worth mentioning is that our latent model can be seen as a *world model* [Ha and Schmidhuber, 2018; Freeman et al., 2019; Hafner et al., 2021]. Indeed, the latent model captures the dynamics of the environment, so it can be used to plan in the latent space. The structure of the latent space enforced in the context of this thesis even ensures its tractability. This further

opens the door to diverse planning approaches (e.g., explicit exploration of the model or through simulations). The main challenge here is to ensure that policies learned entirely in the latent space do not break the abstraction and bisimulation guarantees of the latent model. Indeed, the current approach provides guarantees *with respect to the behavioral policy* — the one used to collect training data. Planning usually goes beyond this policy as it explores states and actions in regions of the latent space that have not been necessarily widely visited through the behavioral policy. In spirit, one can draw connections with the concept of *safe policy improvement* (e.g., Scholl et al., 2022; Simão et al., 2023; Wienhöft et al., 2023). Planning in the latent model while preserving the bisimulation guarantees would ensure that policies learned *in the latent space* can easily be lifted back to the original environment with similar performance, while also improving sample efficiency.

Multi-objective Reasoning

One of the main goals of this thesis was to formally reason about the behaviors induced by deep RL policies. The real world is complex and often involves multiple trade-offs between (possibly conflicting) constraints, costs, rewards, and specifications. In fact, the willingness to optimize a reward signal while ensuring that a set of safety requirements are met is a typical example of a multi-objective problem. Furthermore, formal reasoning also involves the ability to reason about those multiple trade-offs. This is then a natural extension of our work to consider multi-objective reasoning.

Multi-objective deep RL (e.g., Reymond and Nowé, 2019; Abels et al., 2019; Reymond et al., 2022; Hayes et al., 2022; Alegre et al., 2023; Röpke et al., 2024) and model-checking (e.g., Chatterjee et al., 2006; Etessami et al., 2008; Forejt et al., 2012; Hartmanns et al., 2018; Delgrange et al., 2020) have been studied *in parallel* in the literature, but their combination has never been fully addressed.

Many of the deep RL approaches enable formal reasoning by recording in the state space the rewards accumulated on every dimension (one dimension per objective) [Hayes et al., 2022]. Some of them condition a single policy on the desired trade-off or optimization direction (e.g., Abels et al., 2019; Reymond et al., 2022), which can be seen as *goal-conditioned* RL approaches [Liu et al., 2022a]. Our framework naturally applies in such cases: this enables, e.g., checking if some costs remain below a threshold, or if a return is guaranteed on some dimension. Furthermore, for goal-conditioned approaches, one can further check whether the conditioning of the policy is eventually met (for instance, by checking if the latent model can achieve a desired return).

This is the idea we experimented in *Wasserstein auto-encoded Pareto conditioned networks* (WAE-PCN) [Delgrange et al., 2023b]: combining WAE-MDPs with Pareto conditioned networks (PCN, Reymond et al., 2022). PCN offers state-of-the-art performance but struggles when the environment exhibits high stochasticity. WAE-MDP corrects PCN when the latent model detects unexpected events. Moreover, we

adapted the abstraction quality guarantees to the multi-objective setting, ensuring the verifiability of the PCN’s results. However, the way goal-conditioned algorithms are trained is intricate and involves manipulating the replay buffer, continuously changing the distribution of the data collected, which exhibits a high degree of non-stationarity. This yields a challenge for the guarantees provided by the latent model, as they are based on the assumption that the data distribution eventually turns stationary. Future work should focus on fixing those instability issues and may involve, e.g., the use of off-policy learning techniques [Gelada and Bellemare, 2019].

Refining the PAC Guarantees

Finally, from a theoretical standpoint, there are several possibilities for future research, notably for refining the PAC guarantees provided with the latent model. As we already mentioned in Chapter 3, the PAC guarantees that we developed to estimate the transition loss (Lemma 14) are rather conservative. In fact, the theoretical transition loss involves the transition function of the original environment, which is not available in practice when we need to empirically estimate the former. To alleviate this problem, we developed an upper bound on the transition loss, which is learnable via tractable simulations of the environment, but sensitive to the entropy of its dynamics. As a result, the PAC bounds provided may sometimes yield high numerical values even if the latent model is accurate.

We also presented theoretical PAC bounds for value difference and bisimulation distance (Theorem 25). Some of them rely on an intricate termination criterion, which suggests probabilistic algorithms that we did not explicitly develop. Future research may involve the development of a dedicated algorithm to estimate those bounds efficiently, in practice, to provide interpretable results faster.

Bibliography

- Abadi, M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng
2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. Cited on pages 320 and 321.
- Abels, A., D. M. Roijers, T. Lenaerts, A. Nowé, and D. Steckelmacher
2019. Dynamic weights in multi-objective deep reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, K. Chaudhuri and R. Salakhutdinov, eds., volume 97 of *Proceedings of Machine Learning Research*, Pp. 11–20. PMLR. Cited on page 283.
- Achiam, J., D. Held, A. Tamar, and P. Abbeel
2017. Constrained policy optimization. In *ICML*, volume 70, Pp. 22–31. PMLR. Cited on page 244.
- Akiba, T., S. Sano, T. Yanase, T. Ohta, and M. Koyama
2019. Optuna: A next-generation hyperparameter optimization framework. *CoRR*, abs/1907.10902. Cited on page 336.
- Alamdari, P. A., G. Avni, T. A. Henzinger, and A. Lukina
2020. Formal methods with a touch of magic. In *2020 Formal Methods in Computer Aided Design, FMCAD 2020, Haifa, Israel, September 21-24, 2020*, Pp. 138–147. IEEE. Cited on pages 135 and 243.
- Alegre, L. N., A. L. C. Bazzan, D. M. Roijers, A. Nowé, and B. C. da Silva
2023. Sample-efficient multi-objective learning via generalized policy improvement prioritization. In *AAMAS*, N. Agmon, B. An, A. Ricci, and W. Yeoh, eds., Pp. 2003–2012. ACM. Cited on pages 277 and 283.

- Alemi, A. A., B. Poole, I. Fischer, J. V. Dillon, R. A. Saurous, and K. Murphy
2018. Fixing a broken ELBO. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, J. G. Dy and A. Krause, eds., volume 80 of *Proceedings of Machine Learning Research*, Pp. 159–168. PMLR. Cited on pages 164 and 173.
- Alshiekh, M., R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu
2018. Safe reinforcement learning via shielding. In *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*, S. A. McIlraith and K. Q. Weinberger, eds., Pp. 2669–2678. AAAI Press. Cited on pages 135, 244, and 282.
- Alur, R., S. Bansal, O. Bastani, and K. Jothimurugan
2022. A framework for transforming specifications in reinforcement learning. In *Principles of Systems Design - Essays Dedicated to Thomas A. Henzinger on the Occasion of His 60th Birthday*, volume 13660, Pp. 604–624. Springer. Cited on page 244.
- Alur, R., O. Bastani, K. Jothimurugan, M. Perez, F. Somenzi, and A. Trivedi
2023. Policy synthesis and reinforcement learning for discounted LTL. In *Computer Aided Verification - 35th International Conference, CAV 2023, Paris, France, July 17-22, 2023, Proceedings, Part I*, C. Enea and A. Lal, eds., volume 13964 of *Lecture Notes in Computer Science*, Pp. 415–435. Springer. Cited on page 282.
- Amin, S., M. Gomrokchi, H. Satija, H. van Hoof, and D. Precup
2021. A survey of exploration methods in reinforcement learning. *CoRR*, abs/2109.00157. Cited on page 54.
- Andrychowicz, M., D. Crow, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba
2017. Hindsight experience replay. In *NeurIPS*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, eds., Pp. 5048–5058. Cited on page 277.
- Arjovsky, M., S. Chintala, and L. Bottou
2017. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, D. Precup and Y. W. Teh, eds., volume 70 of *Proceedings of Machine Learning Research*, Pp. 214–223. PMLR. Cited on page 234.

Asmussen, S. and P. W. Glynn

2011. A new proof of convergence of mcmc via the ergodic theorem. *Statistics & Probability Letters*, 81(10):1482–1485. Cited on pages 44 and 167.

Athreya, K. B. and P. Ney

1978. A new approach to the limit theory of recurrent markov chains. *Transactions of the American Mathematical Society*, 245:493–501. Cited on page 44.

Avrachenkov, K., A. B. Piunovskiy, and Y. Zhang

2013. Markov processes with restart. *J. Appl. Probab.*, 50(4):960–968. Cited on page 45.

Azar, M. G., I. Osband, and R. Munos

2017. Minimax regret bounds for reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6–11 August 2017*, D. Precup and Y. W. Teh, eds., volume 70 of *Proceedings of Machine Learning Research*, Pp. 263–272. PMLR. Cited on page 57.

Bacci, E. and D. Parker

2020. Probabilistic guarantees for safe deep reinforcement learning. In *Formal Modeling and Analysis of Timed Systems - 18th International Conference, FORMATS 2020, Vienna, Austria, September 1–3, 2020, Proceedings*, N. Bertrand and N. Jansen, eds., volume 12288 of *LNCS*, Pp. 231–248. Springer. Cited on page 135.

Bacci, E. and D. Parker

2022. Verified probabilistic policies for deep reinforcement learning. In *NASA Formal Methods*, J. V. Deshmukh, K. Havelund, and I. Perez, eds., Pp. 193–212, Cham. Springer International Publishing. Cited on page 135.

Badings, T. S., L. Romao, A. Abate, D. Parker, H. A. Poonawala, M. Stoelinga, and N. Jansen

2023. Robust control for dynamical systems with non-Gaussian noise via formal abstractions. *J. Artif. Intell. Res.*, 76:341–391. Cited on page 244.

Baier, C., L. de Alfaro, V. Forejt, and M. Kwiatkowska

2018. Model checking probabilistic systems. In *Handbook of Model Checking*, E. M. Clarke, T. A. Henzinger, H. Veith, and R. Bloem, eds., Pp. 963–999. Springer. Cited on pages 38 and 39.

Baier, C. and J. Katoen

2008. *Principles of model checking*. MIT Press. Cited on pages 17, 38, 39, 42, 78, 86, 87, 88, 92, 122, and 167.

Baier, C., S. Kiefer, J. Klein, D. Müller, and J. Worrell

2023. Markov chains and unambiguous automata. *J. Comput. Syst. Sci.*, 136:113–134. Cited on page 95.

BIBLIOGRAPHY

- Balaji, N., S. Kiefer, P. Novotný, G. A. Pérez, and M. Shirmohammadi
2019. On the complexity of value iteration. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9–12, 2019, Patras, Greece*, C. Baier, I. Chatzigiannakis, P. Flocchini, and S. Leonardi, eds., volume 132 of *LIPIcs*, Pp. 102:1–102:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik. Cited on pages 51 and 53.
- Bargiacchi, E., D. M. Roijers, and A. Nowé
2020. Ai-toolbox: A c++ library for reinforcement learning and planning (with python bindings). *Journal of Machine Learning Research*, 21(102):1–12. Cited on page 53.
- Barto, A. G. and S. Mahadevan
2003. Recent advances in hierarchical reinforcement learning. *Discret. Event Dyn. Syst.*, 13(4):341–379. Cited on page 243.
- Bastani, O., Y. Pu, and A. Solar-Lezama
2018. Verifiable reinforcement learning via policy extraction. In *NeurIPS*, Pp. 2499–2509. Cited on page 243.
- Bazille, H., B. Genest, C. Jégourel, and J. Sun
2020. Global PAC bounds for learning discrete time markov chains. In *Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21–24, 2020, Proceedings, Part II*, S. K. Lahiri and C. Wang, eds., volume 12225 of *Lecture Notes in Computer Science*, Pp. 304–326. Springer. Cited on pages 99 and 100.
- Bellemare, M. G., S. Candido, P. S. Castro, J. Gong, M. C. Machado, S. Moitra, S. S. Ponda, and Z. Wang
2020. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nat.*, 588(7836):77–82. Cited on page 133.
- Bellemare, M. G., W. Dabney, and M. Rowland
2023. *Distributional Reinforcement Learning*. MIT Press. <http://www.distributional-rl.org>. Cited on pages 70, 275, and 340.
- Bellman, R.
1957. *Dynamic Programming*. Dover Publications. Cited on pages 47 and 50.
- Bengio, Y., N. Léonard, and A. C. Courville
2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432. Cited on page 193.
- Bertsekas, D. P. and J. N. Tsitsiklis
1991. An analysis of stochastic shortest path problems. *Math. Oper. Res.*, 16(3):580–595. Cited on pages 59, 76, 84, and 85.

- Bertsekas, D. P. and J. N. Tsitsiklis
1996. *Neuro-dynamic programming*, volume 3 of *Optimization and neural computation series*. Athena Scientific. Cited on page 70.
- Blahoudek, F., M. Kretínský, and J. Strejcek
2013. Comparison of LTL to deterministic rabin automata translators. In *Logic for Programming, Artificial Intelligence, and Reasoning - 19th International Conference, LPAR-19, Stellenbosch, South Africa, December 14-19, 2013. Proceedings*, K. L. McMillan, A. Middeldorp, and A. Voronkov, eds., volume 8312 of *Lecture Notes in Computer Science*, Pp. 164–172. Springer. Cited on page 88.
- Bork, A., J. Katoen, and T. Quatmann
2022. Under-approximating expected total rewards in pomdps. In *Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part II*, D. Fisman and G. Rosu, eds., volume 13244 of *Lecture Notes in Computer Science*, Pp. 22–40. Springer. Cited on page 214.
- Bousquet, O., S. Gelly, I. Tolstikhin, C.-J. Simon-Gabriel, and B. Schölkopf
2017. From optimal transport to generative modeling: the vegan cookbook. *arXiv: Machine Learning*. Cited on page 183.
- Brin, S. and L. Page
1998. The anatomy of a large-scale hypertextual web search engine. *Comput. Networks*, 30(1-7):107–117. Cited on page 45.
- Brockman, G., V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba
2016. Openai gym. *CoRR*, abs/1606.01540. Cited on pages 56 and 176.
- Budde, C. E., A. Hartmanns, M. Klauck, J. Kretínský, D. Parker, T. Quatmann, A. Turrini, and Z. Zhang
2020. On correctness, precision, and performance in quantitative verification - qcomp 2020 competition report. In *Leveraging Applications of Formal Methods, Verification and Validation: Tools and Trends - 9th International Symposium on Leveraging Applications of Formal Methods, ISoLA 2020, Rhodes, Greece, October 20-30, 2020, Proceedings, Part IV*, T. Margaria and B. Steffen, eds., volume 12479 of *Lecture Notes in Computer Science*, Pp. 216–241. Springer. Cited on pages 53 and 176.
- Bulychev, P. E., F. Cassez, A. David, K. G. Larsen, J. Raskin, and P. Reynier
2012. Controllers with minimal observation power (application to timed systems). In *Automated Technology for Verification and Analysis - 10th International Symposium*,

BIBLIOGRAPHY

- ATVA 2012, Thiruvananthapuram, India, October 3-6, 2012. *Proceedings*, S. Chakraborty and M. Mukund, eds., volume 7561 of *Lecture Notes in Computer Science*, Pp. 223–237. Springer. Cited on page 215.
- Burke, M., Y. Hristov, and S. Ramamoorthy
2019. Hybrid system identification using switching density networks. In *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, L. P. Kaelbling, D. Kragic, and K. Sugiura, eds., volume 100 of *Proceedings of Machine Learning Research*, Pp. 172–181. PMLR. Cited on page 173.
- Camacho, A., R. T. Icarte, T. Q. Klassen, R. A. Valenzano, and S. A. McIlraith
2019. LTL and beyond: Formal languages for reward function specification in reinforcement learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, S. Kraus, ed., Pp. 6065–6073. ijcai.org. Cited on pages 101 and 282.
- Carr, S., N. Jansen, and U. Topcu
2020. Verifiable rnn-based policies for pomdps under temporal logic constraints. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, C. Bessiere, ed., Pp. 4121–4127. ijcai.org. Cited on page 135.
- Cassirer, A., G. Barth-Maron, E. Brevdo, S. Ramos, T. Boyd, T. Sottiaux, and M. Kroiss
2021. Reverb: A framework for experience replay. Cited on page 320.
- Castro, P. S., T. Kastner, P. Panangaden, and M. Rowland
2021. Mico: Improved representations via sampling-based state similarity for markov decision processes. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, eds., Pp. 30113–30126. Cited on page 136.
- Ceusters, G., R. C. Rodríguez, A. B. García, R. Franke, G. Deconinck, L. Helsen, A. Nowé, M. Messaïe, and L. R. Camargo
2021. Model-predictive control and reinforcement learning in multi-energy system case studies. *Applied Energy*, 303:117634. Cited on pages 15 and 133.
- Chakraborty, D., D. Busatto-Gaston, J.-F. Raskin, and G. A. Pérez
2023. Formally-sharp dagger for mcts: Lower-latency monte carlo tree search using data aggregation with formal methods. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS '23*, P. 1354–1362, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems. Cited on page 135.

- Chatterjee, K., M. Chmelik, R. Gupta, and A. Kanodia
2016. Optimal cost almost-sure reachability in pomdps. *Artif. Intell.*, 234:26–48. Cited on page 216.
- Chatterjee, K., L. de Alfaro, R. Majumdar, and V. Raman
2010. Algorithms for game metrics (full version). *Log. Methods Comput. Sci.*, 6(3). Cited on page 23.
- Chatterjee, K., R. Majumdar, and T. A. Henzinger
2006. Markov decision processes with multiple objectives. In *STACS 2006, 23rd Annual Symposium on Theoretical Aspects of Computer Science, Marseille, France, February 23–25, 2006, Proceedings*, B. Durand and W. Thomas, eds., volume 3884 of *Lecture Notes in Computer Science*, Pp. 325–336. Springer. Cited on page 283.
- Chen, X., Y. M. Mu, P. Luo, S. Li, and J. Chen
2022. Flow-based recurrent belief state learning for pomdps. In *International Conference on Machine Learning*, Pp. 3444–3468. PMLR. Cited on pages 209 and 212.
- Coppersmith, D. and S. Winograd
1987. Matrix multiplication via arithmetic progressions. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, A. V. Aho, ed., Pp. 1–6. ACM. Cited on page 49.
- Cornel, D. S., W. Gerstner, and J. Brea
2018. Efficient modelbased deep reinforcement learning with variational state tabulation. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10–15, 2018*, J. G. Dy and A. Krause, eds., volume 80 of *Proceedings of Machine Learning Research*, Pp. 1057–1066. PMLR. Cited on pages 136 and 194.
- Daniele, M., F. Giunchiglia, and M. Y. Vardi
1999. Improved automata generation for linear temporal logic. In *Computer Aided Verification*, N. Halbwachs and D. Peled, eds., Pp. 249–260, Berlin, Heidelberg. Springer Berlin Heidelberg. Cited on pages 87 and 88.
- Dann, C. and E. Brunskill
2015. Sample complexity of episodic fixed-horizon reinforcement learning. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7–12, 2015, Montreal, Quebec, Canada*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds., Pp. 2818–2826. Cited on page 57.

BIBLIOGRAPHY

- Dann, C., T. Lattimore, and E. Brunskill
2017. Unifying PAC and regret: Uniform PAC bounds for episodic reinforcement learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, eds., Pp. 5713–5723. Cited on page 57.
- Dantzig, G. B.
1991. *Linear Programming and Extensions*. Princeton University Press. Cited on page 49.
- de Alfaro, L., T. A. Henzinger, and R. Majumdar
2001. From verification to control: Dynamic programs for omega-regular objectives. In *16th Annual IEEE Symposium on Logic in Computer Science, Boston, Massachusetts, USA, June 16-19, 2001, Proceedings*, Pp. 279–290. IEEE Computer Society. Cited on page 113.
- de Alfaro, L., T. A. Henzinger, and R. Majumdar
2003. Discounting the future in systems theory. In *Automata, Languages and Programming, 30th International Colloquium, ICALP 2003, Eindhoven, The Netherlands, June 30 - July 4, 2003. Proceedings*, J. C. M. Baeten, J. K. Lenstra, J. Parrow, and G. J. Woeginger, eds., volume 2719 of *Lecture Notes in Computer Science*, Pp. 1022–1037. Springer. Cited on pages 23, 102, 108, 110, 113, 114, 116, 122, 130, and 131.
- Degrave, J., F. Felici, J. Buchli, M. Neunert, B. D. Tracey, F. Carpanese, T. Ewalds, R. Hafner, A. Abdolmaleki, D. de Las Casas, C. Donner, L. Fritz, C. Galperti, A. Huber, J. Keeling, M. Tsimpoukelli, J. Kay, A. Merle, J. Moret, S. Noury, F. Pesamosca, D. Pfau, O. Sauter, C. Sommariva, S. Coda, B. Duval, A. Fasoli, P. Kohli, K. Kavukcuoglu, D. Hassabis, and M. A. Riedmiller
2022. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nat.*, 602(7897):414–419. Cited on pages 15 and 133.
- d’Epenoux, F.
1963. A probabilistic production and inventory problem. *Management Science*, 10(1):98–108. Cited on page 48.
- Desharnais, J., V. Gupta, R. Jagadeesan, and P. Panangaden
2004. Metrics for labelled markov processes. *Theor. Comput. Sci.*, 318(3):323–354. Cited on pages 23, 130, and 131.
- Dillon, J. V., I. Langmore, D. Tran, E. Brevdo, S. Vasudevan, D. Moore, B. Patton, A. Alemi, M. Hoffman, and R. A. Saurous
2017. Tensorflow distributions. Cited on pages 320 and 321.

- Domingues, O. D., P. Ménard, E. Kaufmann, and M. Valko
2021. Episodic reinforcement learning in finite mdps: Minimax lower bounds revisited. In *Algorithmic Learning Theory, 16-19 March 2021, Virtual Conference, Worldwide*, V. Feldman, K. Ligett, and S. Sabato, eds., volume 132 of *Proceedings of Machine Learning Research*, Pp. 578–598. PMLR. Cited on page 57.
- Dong, Z., B. A. Seybold, K. Murphy, and H. H. Bui
2020. Collapsed amortized variational inference for switching nonlinear dynamical systems. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020*, volume 119 of *Proceedings of Machine Learning Research*, Pp. 2638–2647. PMLR. Cited on page 173.
- Dudley, R. M.
2002. *Real Analysis and Probability*, Cambridge Studies in Advanced Mathematics, 2 edition. Cambridge University Press. Cited on page 167.
- Durrett, R.
2010. *Probability: Theory and Examples, 4th Edition*. Cambridge University Press. Cited on page 44.
- Dyer, J., N. Bishop, Y. Felekis, F. M. Zennaro, A. Calinescu, T. Damoulas, and M. Wooldridge
2023. Interventionally consistent surrogates for agent-based simulators. Cited on page 136.
- Emerson, E. A. and C. S. Jutla
1991. Tree automata, mu-calculus and determinacy (extended abstract). In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*, Pp. 368–377. IEEE Computer Society. Cited on pages 109 and 113.
- Ernst, D., P. Geurts, and L. Wehenkel
2005. Tree-based batch mode reinforcement learning. *JMLR*, 6(Apr):503–556. Cited on page 243.
- Esparza, J., J. Kretínský, J. Raskin, and S. Sickert
2017. From LTL and limit-deterministic büchi automata to deterministic parity automata. In *Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings, Part I*, A. Legay and T. Margaria, eds., volume 10205 of *Lecture Notes in Computer Science*, Pp. 426–442. Cited on page 113.

- Etessami, K., M. Z. Kwiatkowska, M. Y. Vardi, and M. Yannakakis
 2008. Multi-objective model checking of markov decision processes. *Log. Methods Comput. Sci.*, 4(4). Cited on page 283.
- Fajtl, J., V. Argyriou, D. Monekosso, and P. Remagnino
 2020. Latent bernoulli autoencoder. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, Pp. 2964–2974. PMLR. Cited on page 193.
- Fan, J., Z. Wang, Y. Xie, and Z. Yang
 2020. A theoretical analysis of deep q-learning. In *Proceedings of the 2nd Annual Conference on Learning for Dynamics and Control, L4DC 2020, Online Event, Berkeley, CA, USA, 11-12 June 2020*, A. M. Bayen, A. Jadabaie, G. J. Pappas, P. A. Parrilo, B. Recht, C. J. Tomlin, and M. N. Zeilinger, eds., volume 120 of *Proceedings of Machine Learning Research*, Pp. 486–489. PMLR. Cited on pages 66 and 70.
- Farrington, J., K. Li, W. K. Wong, and M. Utley
 2023. Going faster to see further: Gpu-accelerated value iteration and simulation for perishable inventory control using JAX. *CoRR*, abs/2303.10672. Cited on page 53.
- Ferns, N., P. Panangaden, and D. Precup
 2005. Metrics for markov decision processes with infinite state spaces. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*, Pp. 201–208. AUAI Press. Cited on pages 23, 129, and 131.
- Ferns, N., P. Panangaden, and D. Precup
 2011. Bisimulation metrics for continuous markov decision processes. *SIAM J. Comput.*, 40(6):1662–1714. Cited on page 23.
- Forejt, V., M. Z. Kwiatkowska, G. Norman, and D. Parker
 2011. Automated verification techniques for probabilistic systems. In *Formal Methods for Eternal Networked Software Systems - 11th International School on Formal Methods for the Design of Computer, Communication and Software Systems, SFM 2011, Bertinoro, Italy, June 13-18, 2011. Advanced Lectures*, M. Bernardo and V. Issarny, eds., volume 6659 of *Lecture Notes in Computer Science*, Pp. 53–113. Springer. Cited on pages 83 and 85.
- Forejt, V., M. Z. Kwiatkowska, and D. Parker
 2012. Pareto curves for probabilistic model checking. In *Automated Technology for Verification and Analysis - 10th International Symposium, ATVA 2012, Thiruvananthapuram, India, October 3-6, 2012. Proceedings*, S. Chakraborty and M. Mukund, eds., volume 7561 of *Lecture Notes in Computer Science*, Pp. 317–332. Springer. Cited on page 283.

- François-Lavet, V., Y. Bengio, D. Precup, and J. Pineau
2019. Combined reinforcement learning via abstract representations. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, Pp. 3582–3589. AAAI Press. Cited on page 136.
- François-Lavet, V., Y. Bengio, D. Precup, and J. Pineau
2019. Combined reinforcement learning via abstract representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, Pp. 3582–3589. Cited on page 233.
- Freeman, C. D., D. Ha, and L. Metz
2019. Learning to predict without looking ahead: World models without forward prediction. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, eds., Pp. 5380–5391. Cited on pages 136 and 282.
- Fu, J. and U. Topcu
2014. Probably approximately correct MDP learning and control with temporal logic constraints. In *Robotics: Science and Systems X*. Cited on page 244.
- Gelada, C. and M. G. Bellemare
2019. Off-policy deep reinforcement learning by bootstrapping the covariate shift. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, Pp. 3647–3655. AAAI Press. Cited on pages 76 and 284.
- Gelada, C., S. Kumar, J. Buckman, O. Nachum, and M. G. Bellemare
2019. Deepmdp: Learning continuous latent space models for representation learning. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, K. Chaudhuri and R. Salakhutdinov, eds., volume 97 of *Proceedings of Machine Learning Research*, Pp. 2170–2179. PMLR. Cited on pages 136, 138, 139, 145, 146, 153, and 233.
- Genevay, A.
2019. *Entropy-Regularized Optimal Transport for Machine Learning. (Régularisation Entropique du Transport Optimal pour le Machine Learning)*. PhD thesis, PSL Research University, Paris, France. Cited on pages 139 and 153.

- Geng, X.
2023. *Lecture Notes on Advanced Probability*. The University of Melbourne. Cited on page 168.
- Germain, M., K. Gregor, I. Murray, and H. Larochelle
2015. MADE: masked autoencoder for distribution estimation. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, F. R. Bach and D. M. Blei, eds., volume 37 of *JMLR Workshop and Conference Proceedings*, Pp. 881–889. JMLR.org. Cited on page 195.
- Giacobbe, M., M. Hasanbeig, D. Kroening, and H. Wijk
2021. Shielding atari games with bounded prescience. In *AAMAS*, Pp. 1507–1509. ACM. Cited on page 244.
- Givan, R., T. L. Dean, and M. Greig
2003. Equivalence notions and model minimization in markov decision processes. *Artif. Intell.*, 147(1-2):163–223. Cited on pages 18, 122, and 224.
- Grädel, E., W. Thomas, and T. Wilke, eds.
2002. *Automata Logics, and Infinite Games: A Guide to Current Research*. Berlin, Heidelberg: Springer-Verlag. Cited on page 87.
- Granzotto, M., R. Postoyan, D. Nesic, L. Busoniu, and J. Daafouz
2021. When to stop value iteration: stability and near-optimality versus computation. In *Proceedings of the 3rd Annual Conference on Learning for Dynamics and Control, L4DC 2021, 7-8 June 2021, Virtual Event, Switzerland*, A. Jadababae, J. Lygeros, G. J. Pappas, P. A. Parrilo, B. Recht, C. J. Tomlin, and M. N. Zeilinger, eds., volume 144 of *Proceedings of Machine Learning Research*, Pp. 412–424. PMLR. Cited on page 51.
- Gregor, K., G. Papamakarios, F. Besse, L. Buesing, and T. Weber
2018. Temporal Difference Variational Auto-Encoder. *7th International Conference on Learning Representations, ICLR 2019*. Cited on pages 209 and 210.
- Gregor, K., D. J. Rezende, F. Besse, Y. Wu, H. Merzic, and A. van den Oord
2019. Shaping Belief States with Generative Environment Models for RL. *Advances in Neural Information Processing Systems*, 32. Cited on page 209.
- Guadarrama, S., A. Korattikara, O. Ramirez, P. Castro, E. Holly, S. Fishman, K. Wang, E. Gonina, N. Wu, E. Kokiopoulou, L. Sbaiz, J. Smith, G. Bartók, J. Berent, C. Harris, V. Vanhoucke, and E. Brevdo
2018. TF-Agents: A library for reinforcement learning in tensorflow. <https://github.com/tensorflow/agents>. [Online; accessed 25-June-2019]. Cited on pages 320 and 321.

- Gulrajani, I., F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville
2017. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, eds., Pp. 5767–5777. Cited on pages 191, 197, 234, and 235.
- Ha, D. and J. Schmidhuber
2018. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems 31*, Pp. 2451–2463. Curran Associates, Inc. <https://worldmodels.github.io>. Cited on pages 136 and 282.
- Haarnoja, T., B. Moran, G. Lever, S. H. Huang, D. Tirumala, J. Humplik, M. Wulfmeier, S. Tunyasuvunakool, N. Y. Siegel, R. Hafner, M. Bloesch, K. Hartikainen, A. Byravan, L. Hasenclever, Y. Tassa, F. Sadeghi, N. Batchelor, F. Casarini, S. Saliceti, C. Game, N. Sreendra, K. Patel, M. Gwira, A. Huber, N. Hurley, F. Nori, R. Hadsell, and N. Heess
2024. Learning agile soccer skills for a bipedal robot with deep reinforcement learning. *Science Robotics*, 9(89):eadi8022. Cited on page 15.
- Haarnoja, T., A. Zhou, P. Abbeel, and S. Levine
2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, J. G. Dy and A. Krause, eds., volume 80 of *Proceedings of Machine Learning Research*, Pp. 1856–1865. PMLR. Cited on pages 68, 73, and 176.
- Haddad, S. and B. Monmege
2014. Reachability in mdps: Refining convergence of value iteration. In *Reachability Problems - 8th International Workshop, RP 2014, Oxford, UK, September 22-24, 2014. Proceedings*, J. Ouaknine, I. Potapov, and J. Worrell, eds., volume 8762 of *Lecture Notes in Computer Science*, Pp. 125–137. Springer. Cited on page 86.
- Hafner, D., T. P. Lillicrap, J. Ba, and M. Norouzi
2020. Dream to control: Learning behaviors by latent imagination. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. Cited on page 209.
- Hafner, D., T. P. Lillicrap, M. Norouzi, and J. Ba
2021. Mastering atari with discrete world models. In *International Conference on Learning Representations*. Cited on pages 136, 193, 209, 210, 239, and 282.
- Hartmanns, A., S. Junges, J. Katoen, and T. Quatmann
2018. Multi-cost bounded reachability in MDP. In *Tools and Algorithms for the*

- Construction and Analysis of Systems - 24th International Conference, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings, Part II*, D. Beyer and M. Huisman, eds., volume 10806 of *Lecture Notes in Computer Science*, Pp. 320–339. Springer. Cited on page 283.
- Hartmanns, A., S. Junges, T. Quatmann, and M. Weininger
2023. A practitioner’s guide to MDP model checking algorithms. In *Tools and Algorithms for the Construction and Analysis of Systems - 29th International Conference, TACAS 2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Paris, France, April 22-27, 2023, Proceedings, Part I*, S. Sankaranarayanan and N. Sharygina, eds., volume 13993 of *Lecture Notes in Computer Science*, Pp. 469–488. Springer. Cited on page 53.
- Hartmanns, A. and B. L. Kaminski
2020. Optimistic value iteration. In *Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part II*, S. K. Lahiri and C. Wang, eds., volume 12225 of *Lecture Notes in Computer Science*, Pp. 488–511. Springer. Cited on pages 51, 85, and 86.
- Hasanbeig, M., A. Abate, and D. Kroening
2020. Cautious reinforcement learning with logical constraints. In *AAMAS*, Pp. 483–491. International Foundation for Autonomous Agents and Multiagent Systems. Cited on page 244.
- Hasanbeig, M., N. Y. Jeppu, A. Abate, T. Melham, and D. Kroening
2021. Deepsynth: Automata synthesis for automatic task segmentation in deep reinforcement learning. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, Pp. 7647–7656. AAAI Press. Cited on page 135.
- Hausknecht, M. and P. Stone
2015. Deep recurrent q-learning for partially observable MDPs. In *AAAI Fall Symposium - Technical Report*, volume FS-15-06, Pp. 29–37. AI Access Foundation. Cited on page 209.
- Hauskrecht, M.
2000. Value-function approximations for partially observable markov decision processes. *J. Artif. Intell. Res.*, 13:33–94. Cited on page 231.
- Hayes, C. F., R. Radulescu, E. Bargiacchi, J. Källström, M. Macfarlane, M. Reymond, T. Verstraeten, L. M. Zintgraf, R. Dazeley, F. Heintz, E. Howley, A. A. Irissappane,

- P. Mannion, A. Nowé, G. de Oliveira Ramos, M. Restelli, P. Vamplew, and D. M. Roijers
2022. A practical guide to multi-objective reinforcement learning and planning. *Auton. Agents Multi Agent Syst.*, 36(1):26. Cited on page 283.
- Hensel, C., S. Junges, J. Katoen, T. Quatmann, and M. Volk
2022. The probabilistic model checker storm. *Int. J. Softw. Tools Technol. Transf.*, 24(4):589–610. Cited on page 53.
- Hessel, M., J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. G. Azar, and D. Silver
2018. Rainbow: Combining improvements in deep reinforcement learning. In *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*, S. A. McIlraith and K. Q. Weinberger, eds., Pp. 3215–3222. AAAI Press. Cited on pages 70 and 275.
- Hoeffding, W.
1963. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30. Cited on pages 155, 156, and 157.
- Hoffman, M. D., D. M. Blei, C. Wang, and J. W. Paisley
2013. Stochastic variational inference. *J. Mach. Learn. Res.*, 14(1):1303–1347. Cited on page 164.
- Huang, B.
2020. Steady state analysis of episodic reinforcement learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds. Cited on pages 56, 61, 62, 63, 64, 72, and 324.
- Huang, R., Y. Liang, and J. Yang
2024. Provably efficient UCB-type algorithms for learning predictive state representations. In *The Twelfth International Conference on Learning Representations*. Cited on page 212.
- Igl, M., L. Zintgraf, T. A. Le, F. Wood, and S. Whiteson
2018. Deep variational reinforcement learning for POMDPs. In *35th International Conference on Machine Learning, ICML 2018*, volume 5. Cited on pages 209, 212, and 236.
- Jaakkola, T. S., M. I. Jordan, and S. Singh
1993. Convergence of stochastic iterative dynamic programming algorithms. In *Advances in Neural Information Processing Systems 6, [7th NIPS Conference, Denver,*

BIBLIOGRAPHY

- Colorado, USA, 1993], J. D. Cowan, G. Tesauro, and J. Alspector, eds., Pp. 703–710. Morgan Kaufmann. Cited on page 65.
- Jang, E., S. Gu, and B. Poole
2017. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. Cited on pages 170 and 171.
- Janner, M., J. Fu, M. Zhang, and S. Levine
2019. When to trust your model: Model-based policy optimization. In *NeurIPS*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. B. Fox, and R. Garnett, eds., Pp. 12498–12509. Cited on page 277.
- Jansen, N., B. Könighofer, S. Junges, A. Serban, and R. Bloem
2020. Safe Reinforcement Learning Using Probabilistic Shields (Invited Paper). In *31st International Conference on Concurrency Theory (CONCUR 2020)*, I. Konnov and L. Kovács, eds., volume 171 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Pp. 3:1–3:16, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. Cited on pages 135 and 282.
- Jensen, J. L. W. V.
1906. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Mathematica*, 30:175–193. Cited on pages 144, 228, 329, 330, 334, 335, and 336.
- Jothimurugan, K., S. Bansal, O. Bastani, and R. Alur
2021a. Compositional reinforcement learning from logical specifications. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, eds., Pp. 10026–10039. Cited on page 282.
- Jothimurugan, K., O. Bastani, and R. Alur
2021b. Abstract value iteration for hierarchical reinforcement learning. In *AISTATS*, volume 130, Pp. 1162–1170. PMLR. Cited on page 244.
- Junges, S., N. Jansen, C. Dehnert, U. Topcu, and J. Katoen
2016. Safety-constrained reinforcement learning for mdps. In *Tools and Algorithms for the Construction and Analysis of Systems - 22nd International Conference, TACAS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*, M. Chechik and J. Raskin, eds., volume 9636 of *LNCS*, Pp. 130–146. Springer. Cited on page 135.
- Junges, S. and M. T. J. Spaan
2022. Abstraction-refinement for hierarchical probabilistic models. In *CAV*, volume 13371 of *LNCS*, Pp. 102–123. Springer. Cited on page 242.

- Kaelbling, L. P., M. L. Littman, and A. R. Cassandra
1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134. Cited on pages 20, 209, and 214.
- Kantorovich, L. and G. S. Rubinstein
1958. On a space of totally additive functions. *Vestnik Leningrad. Univ*, 13:52–59. Cited on page 126.
- Kazak, Y., C. W. Barrett, G. Katz, and M. Schapira
2019. Verifying deep-RL-driven systems. In *Proc. of NetAI@SIGCOMM*, Pp. 83–89. Cited on page 243.
- Kearns, M. J., Y. Mansour, and A. Y. Ng
2002. A sparse sampling algorithm for near-optimal planning in large markov decision processes. *Mach. Learn.*, 49(2-3):193–208. Cited on page 153.
- Kingma, D. P. and J. Ba
2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, eds. Cited on pages 56, 320, and 324.
- Kingma, D. P. and M. Welling
2014. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Y. Bengio and Y. LeCun, eds. Cited on pages 161 and 163.
- Kini, D. and M. Viswanathan
2017a. Complexity of model checking mdps against LTL specifications. In *37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2017, December 11-15, 2017, Kanpur, India*, S. V. Lokam and R. Ramanujam, eds., volume 93 of *LIPIcs*, Pp. 35:1–35:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik. Cited on page 95.
- Kini, D. and M. Viswanathan
2017b. Optimal translation of LTL to limit deterministic automata. In *Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings, Part II*, A. Legay and T. Margaria, eds., volume 10206 of *Lecture Notes in Computer Science*, Pp. 113–129. Cited on page 95.
- Könighofer, B., R. Bloem, R. Ehlers, and C. Pek
2022. Correct-by-construction runtime enforcement in AI - A survey. In *Principles of*

- Systems Design - Essays Dedicated to Thomas A. Henzinger on the Occasion of His 60th Birthday*, volume 13660, Pp. 650–663. Springer. Cited on page 244.
- Kretínský, J., T. Meggendorfer, S. Sickert, and C. Ziegler
2018. Rabinizer 4: From LTL to your favourite deterministic automaton. In *Computer Aided Verification - 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part I*, H. Chockler and G. Weissenbacher, eds., volume 10981 of *Lecture Notes in Computer Science*, Pp. 567–577. Springer. Cited on pages 88 and 113.
- Kretínský, J., T. Meggendorfer, and M. Weininger
2023. Stopping criteria for value iteration on stochastic games with quantitative objectives. In *LICS*, Pp. 1–14. Cited on page 51.
- Krishnamurthy, B. and J. Rexford
2001. *Web Protocols and Practice - HTTP/1.1, Networking Protocols, Caching, and Traffic Measurement*. Addison-Wesley. Cited on page 45.
- Krishnan, S. C., A. Puri, R. K. Brayton, and P. Varaiya
1995. The rabin index and chain automata, with applications to automatas and games. In *Computer Aided Verification, 7th International Conference, Liège, Belgium, July, 3-5, 1995, Proceedings*, P. Wolper, ed., volume 939 of *Lecture Notes in Computer Science*, Pp. 253–266. Springer. Cited on page 112.
- Kuba, J. G., C. A. S. de Witt, and J. N. Foerster
2022. Mirror learning: A unifying framework of policy optimisation. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato, eds., volume 162 of *Proceedings of Machine Learning Research*, Pp. 7825–7844. PMLR. Cited on pages 70 and 75.
- Kulkarni, T. D., K. Narasimhan, A. Saeedi, and J. Tenenbaum
2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *NeurIPS*, Pp. 3675–3683. Cited on page 244.
- Kwiatkowska, M. Z. and D. Parker
2013. Automated verification and strategy synthesis for probabilistic systems. In *Automated Technology for Verification and Analysis - 11th International Symposium, ATVA 2013, Hanoi, Vietnam, October 15-18, 2013. Proceedings*, D. V. Hung and M. Ogawa, eds., volume 8172 of *Lecture Notes in Computer Science*, Pp. 5–22. Springer. Cited on page 40.

- Ladosz, P., L. Weng, M. Kim, and H. Oh
2022. Exploration in deep reinforcement learning: A survey. *Inf. Fusion*, 85:1–22. Cited on page 54.
- Lambrechts, G., A. Bolland, and D. Ernst
2023. Informed POMDP: Leveraging additional information in model-based RL. In *ICML Workshop on New Frontiers in Learning, Control, and Dynamical Systems*. Cited on pages 212 and 239.
- Larsen, K. G. and A. Skou
1991. Bisimulation through probabilistic testing. *Inf. Comput.*, 94(1):1–28. Cited on pages 18 and 122.
- Lauri, M., D. Hsu, and J. Pajarinen
2023. Partially observable markov decision processes in robotics: A survey. *IEEE Transactions on Robotics*, 39(1):21–40. Cited on page 207.
- LeCun, Y., B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel
1989. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1(4):541–551. Cited on page 274.
- Lee, A. X., A. Nagabandi, P. Abbeel, and S. Levine
2020. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6–12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds. Cited on pages 136, 210, and 212.
- Lee, J., A. Agarwal, C. Dann, and T. Zhang
2023. Learning in pomdps is sample-efficient with hindsight observability. In *International Conference on Machine Learning*, Pp. 18733–18773. PMLR. Cited on page 212.
- Libin, P. J. K., A. Moonens, T. Verstraeten, F. Perez-Sanjines, N. Hens, P. Lemey, and A. Nowé
2020. Deep reinforcement learning for large-scale epidemic control. In *Machine Learning and Knowledge Discovery in Databases. Applied Data Science and Demo Track - European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part V*, Y. Dong, G. Ifrim, D. Mladenec, C. Saunders, and S. V. Hoecke, eds., volume 12461 of *Lecture Notes in Computer Science*, Pp. 155–170. Springer. Cited on pages 15 and 133.

BIBLIOGRAPHY

- Lin, L. J.
1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Mach. Learn.*, 8:293–321. Cited on page 68.
- Littman, M. L., T. L. Dean, and L. P. Kaelbling
1995. On the complexity of solving markov decision problems. In *UAI '95: Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence, Montreal, Quebec, Canada, August 18-20, 1995*, P. Besnard and S. Hanks, eds., Pp. 394–402. Morgan Kaufmann. Cited on pages 51 and 53.
- Liu, M., M. Zhu, and W. Zhang
2022a. Goal-conditioned reinforcement learning: Problems and solutions. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, L. D. Raedt, ed., Pp. 5502–5511. ijcai.org. Cited on pages 242 and 283.
- Liu, Q., A. Chung, C. Szepesvári, and C. Jin
2022b. When is partially observable reinforcement learning not scary? In *Conference on Learning Theory, 2-5 July 2022, London, UK*, P. Loh and M. Raginsky, eds., volume 178 of *Proceedings of Machine Learning Research*, Pp. 5175–5220. PMLR. Cited on page 212.
- Lovász, L. and P. Winkler
1995. Exact mixing in an unknown markov chain. *Electron. J. Comb.*, 2. Cited on pages 61 and 154.
- Lu, C., P. Ball, J. Parker-Holder, M. Osborne, and S. J. Roberts
2022. Revisiting design choices in offline model based reinforcement learning. In *ICLR*. Cited on page 277.
- Ma, X., P. Karkus, D. Hsu, and W. S. Lee
2020. Particle filter recurrent neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, Pp. 5101–5108. Cited on page 209.
- Maddison, C. J., A. Mnih, and Y. W. Teh
2017. The concrete distribution: A continuous relaxation of discrete random variables. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. Cited on pages 170, 171, 172, and 324.
- Mason, K. and S. Grijalva
2019. A review of reinforcement learning for autonomous building energy management. *Comput. Electr. Eng.*, 78:300–312. Cited on pages 15 and 133.

Maurer, S. M. and B. A. Huberman

2001. Restart strategies and internet congestion. *Journal of Economic Dynamics and Control*, 25(3):641–654. Agent-based Computational Economics (ACE). Cited on page 45.

Meyn, S. P. and R. L. Tweedie

1993. *Markov Chains and Stochastic Stability*, Communications and Control Engineering Series. Springer. Cited on page 43.

Mnih, V., A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu

2016. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, M. Balcan and K. Q. Weinberger, eds., volume 48 of *JMLR Workshop and Conference Proceedings*, Pp. 1928–1937. JMLR.org. Cited on pages 73 and 212.

Mnih, V., K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis

2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533. Cited on pages 15, 16, 66, 68, 69, 176, and 236.

Monge, G.

1781. Mémoire sur la théorie des déblais et des remblais. In *Histoire de l'Académie royale des sciences avec les mémoires de mathématique et de physique tirés des registres de cette Académie*, Pp. 666 – 705. Cited on page 125.

Morad, S. D., R. Kortvelesy, M. Bettini, S. Liwicki, and A. Prorok

2023. Popgym: Benchmarking partially observable reinforcement learning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net. Cited on page 236.

Murphy, K. P.

2023. *Probabilistic Machine Learning: Advanced Topics*. MIT Press. Cited on page 162.

Müller, A.

1997. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429–443. Cited on pages 127 and 327.

Nair, V. and G. E. Hinton

2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, J. Fürnkranz and T. Joachims, eds., Pp. 807–814. Omnipress. Cited on page 70.

BIBLIOGRAPHY

- Nayak, S. P., L. N. Egidio, M. D. Rossa, A. Schmuck, and R. M. Jungers
2023. Context-triggered abstraction-based control design. *IEEE Open Journal of Control Systems*, 2:277–296. Cited on page 244.
- Ng, A. Y., D. Harada, and S. Russell
1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, I. Bratko and S. Dzeroski, eds., Pp. 278–287. Morgan Kaufmann. Cited on page 274.
- Nishiyama, Y., A. Boularias, A. Gretton, and K. Fukumizu
2012. Hilbert space embeddings of pomdps. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, UAI’12, P. 644–653, Arlington, Virginia, USA. AUAI Press. Cited on page 212.
- Nota, C. and P. S. Thomas
2020. Is the policy gradient a gradient? In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS ’20, Auckland, New Zealand, May 9–13, 2020, A. E. F. Seghrouchni, G. Sukthankar, B. An, and N. Yorke-Smith, eds., Pp. 939–947. International Foundation for Autonomous Agents and Multiagent Systems. Cited on page 75.
- Nowé, A.
1994. *The synthesis of ‘safe’ fuzzy controllers based on reinforcement learning*. PhD thesis, Vrije Universiteit Brussel. Cited on page 65.
- O’Cinneide, C. A.
1993. Entrywise perturbation theory and error analysis for Markov chains. *Numerische Mathematik*, 65(1):109–120. Cited on page 272.
- Oliehoek, F. A., M. T. Spaan, and N. Vlassis
2008. Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research*, 32:289–353. Cited on page 210.
- Papamakarios, G., I. Murray, and T. Pavlakou
2017. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4–9, 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, eds., Pp. 2338–2347. Cited on pages 194, 196, and 232.
- Pardo, F., A. Tavakoli, V. Levдик, and P. Kormushev
2018. Time limits in reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden*,

- July 10-15, 2018, J. G. Dy and A. Krause, eds., volume 80 of *Proceedings of Machine Learning Research*, Pp. 4042–4051. PMLR. Cited on page 56.
- Pascanu, R., T. Mikolov, and Y. Bengio
2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, Pp. 1310–1318. JMLR.org. Cited on page 233.
- Pathak, S., E. Ábrahám, N. Jansen, A. Tacchella, and J. Katoen
2015. A greedy approach for the efficient repair of stochastic models. In *NASA Formal Methods - 7th International Symposium, NFM 2015, Pasadena, CA, USA, April 27-29, 2015, Proceedings*, K. Havelund, G. J. Holzmann, and R. Joshi, eds., volume 9058 of *Lecture Notes in Computer Science*, Pp. 295–309. Springer. Cited on page 281.
- Pinsker, M. S.
1964. *Information and information stability of random variables and processes / by M.S. Pinsker. Translated and edited by Amiel Feinstein*, Holden-Day series in time series analysis. San Francisco: Holden-Day. Cited on pages 127, 169, and 234.
- Pnueli, A.
1977. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*, Pp. 46–57. IEEE Computer Society. Cited on pages 181 and 202.
- Polyak, B. T. and A. B. Juditsky
1992. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855. Cited on page 68.
- Propp, J. G. and D. B. Wilson
1998. How to get a perfectly random sample from a generic markov chain and generate a random spanning tree of a directed graph. *J. Algorithms*, 27(2):170–217. Cited on pages 61 and 154.
- Puterman, M. L.
1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley Series in Probability and Statistics. Wiley. Cited on pages 47, 50, 51, 52, 59, 71, 76, and 84.
- Quatmann, T. and J. Katoen
2018. Sound value iteration. In *Computer Aided Verification - 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part I*, H. Chockler and G. Weissenbacher, eds.,

BIBLIOGRAPHY

- volume 10981 of *Lecture Notes in Computer Science*, Pp. 643–661. Springer. Cited on page 86.
- Raschka, S., J. Patterson, and C. Nolet
2020. Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. *arXiv preprint arXiv:2002.04803*. Cited on page 337.
- Revuz, D.
1984. *Markov Chains*, North-Holland mathematical library, second (revised) edition. Elsevier Science Publishers B.V. Cited on pages 38 and 44.
- Reymond, M., E. Bargiacchi, and A. Nowé
2022. Pareto conditioned networks. In *21st International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2022, Auckland, New Zealand, May 9-13, 2022*, P. Faliszewski, V. Mascardi, C. Pelachaud, and M. E. Taylor, eds., Pp. 1110–1118. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS). Cited on page 283.
- Reymond, M., C. F. Hayes, L. Willem, R. Rădulescu, S. Abrams, D. M. Roijers, E. Howley, P. Mannion, N. Hens, A. Nowé, and P. Libin
2024. Exploring the pareto front of multi-objective covid-19 mitigation policies using reinforcement learning. *Expert Systems with Applications*, 249:123686. Cited on pages 15 and 133.
- Reymond, M. and A. Nowé
2019. Pareto-dqn: Approximating the pareto front in complex multi-objective decision problems. In *Proceedings of the adaptive and learning agents workshop (ALA-19) at AAMAS*. Cited on page 283.
- Robbins, H. and S. Monro
1951. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400 – 407. Cited on pages 55 and 65.
- Robbins, H. and D. Siegmund
1971. A convergence theorem for non negative almost supermartingales and some applications. In *Optimizing Methods in Statistics*, J. S. Rustagi, ed., Pp. 233–257. Academic Press. Cited on page 56.
- Robert B. Ash, C. A. D.-D.
2000. *Probability and Measure Theory*. Harcourt/Academic Press. Cited on page 37.
- Roderick, M., C. Grimm, and S. Tellex
2018. Deep abstract Q-networks. In *AAMAS*, Pp. 131–138. Cited on page 244.

- Röpke, W., M. Reymond, P. Mannion, D. M. Roijers, A. Nowé, and R. Radulescu
2024. Divide and conquer: Provably unveiling the pareto front with multi-objective reinforcement learning. *CoRR*, abs/2402.07182. Cited on page 283.
- Ryzhyk, L., P. Chubb, I. Kuz, E. L. Sueur, and G. Heiser
2009. Automatic device driver synthesis with termite. In *SOSP*, Pp. 73–86. ACM. Cited on page 243.
- Sadigh, D., E. S. Kim, S. Coogan, S. S. Sastry, and S. A. Seshia
2014. A learning based approach to control synthesis of markov decision processes for linear temporal logic specifications. In *53rd IEEE Conference on Decision and Control, CDC 2014, Los Angeles, CA, USA, December 15-17, 2014*, Pp. 1091–1096. IEEE. Cited on page 282.
- Schaul, T., J. Quan, I. Antonoglou, and D. Silver
2016. Prioritized experience replay. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, eds. Cited on pages 70, 174, 175, 275, 320, and 340.
- Schmidhuber, J.
2015. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117. Cited on page 55.
- Scholl, P., F. Dietrich, C. Otte, and S. Udluft
2022. Safe policy improvement approaches and their limitations. In *Agents and Artificial Intelligence - 14th International Conference, ICAART 2022, Virtual Event, February 3-5, 2022, Revised Selected Papers*, A. P. Rocha, L. Steels, and H. J. van den Herik, eds., volume 13786 of *Lecture Notes in Computer Science*, Pp. 74–98. Springer. Cited on page 283.
- Schulman, J., S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz
2015. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, F. R. Bach and D. M. Blei, eds., volume 37 of *JMLR Workshop and Conference Proceedings*, Pp. 1889–1897. JMLR.org. Cited on page 75.
- Schulman, J., P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel
2016. High-dimensional continuous control using generalized advantage estimation. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, eds. Cited on pages 74 and 75.
- Schulman, J., F. Wolski, P. Dhariwal, A. Radford, and O. Klimov
2017. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347. Cited on page 75.

BIBLIOGRAPHY

Serfozo, R.

2009. *Basics of Applied Stochastic Processes*, Probability and Its Applications. Springer Berlin Heidelberg. Cited on pages 72 and 149.

SeyedSalehi, E., N. Akbarzadeh, A. Sinha, and A. Mahajan

2023. Approximate information state based convergence analysis of recurrent q-learning. In *Sixteenth European Workshop on Reinforcement Learning*. Cited on page 212.

Shao, D. and M. Kwiatkowska

2023. Sample efficient model-free reinforcement learning from LTL specifications with optimality guarantees. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, Pp. 4180–4189. ijcai.org. Cited on page 282.

Shekhovtsov, A.

2021. Bias-variance tradeoffs in single-sample binary gradient estimators. In *Pattern Recognition - 43rd DAGM German Conference, DAGM GCPR 2021, Bonn, Germany, September 28 - October 1, 2021, Proceedings*, C. Bauckhage, J. Gall, and A. G. Schwing, eds., volume 13024 of *Lecture Notes in Computer Science*, Pp. 127–141. Springer. Cited on page 193.

Sickert, S., J. Esparza, S. Jaax, and J. Kretínský

2016. Limit-deterministic büchi automata for linear temporal logic. In *Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part II*, S. Chaudhuri and A. Farzan, eds., volume 9780 of *Lecture Notes in Computer Science*, Pp. 312–332. Springer. Cited on page 95.

Simão, T. D., N. Jansen, and M. T. J. Spaan

2021. Always safe: Reinforcement learning without safety constraint violations during training. In *AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3-7, 2021*, F. Dignum, A. Lomuscio, U. Endriss, and A. Nowé, eds., Pp. 1226–1235. ACM. Cited on page 135.

Simão, T. D., M. Suilen, and N. Jansen

2023. Safe policy improvement for pomdps via finite-state controllers. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, B. Williams, Y. Chen, and J. Neville, eds., Pp. 15109–15117. AAAI Press. Cited on page 283.

- Smallwood, R. D. and E. J. Sondik
1973. The optimal control of partially observable markov processes over a finite horizon. *Oper. Res.*, 21(5):1071–1088. Cited on page 231.
- Sondik, E. J.
1978. The optimal control of partially observable markov processes over the infinite horizon: Discounted costs. *Oper. Res.*, 26(2):282–304. Cited on page 231.
- Sootla, A., A. I. Cowen-Rivers, T. Jafferjee, Z. Wang, D. H. Mguni, J. Wang, and H. Ammar
2022. Sauté RL: Almost surely safe reinforcement learning using state augmentation. In *ICML*, volume 162, Pp. 20423–20443. PMLR. Cited on page 244.
- Sriperumbudur, B. K., K. Fukumizu, A. Gretton, B. Schölkopf, and G. R. Lanckriet
2009. On integral probability metrics, ϕ -divergences and binary classification. *arXiv preprint arXiv:0901.2698*. Cited on pages 127 and 327.
- Staessens, Tom
2023. *Bringing reinforcement learning to real-world mechatronic systems : a hybrid learning control approach*. PhD thesis, Ghent University. Cited on page 215.
- Stein, E. M. and R. Shakarchi
2005. *Real Analysis: Measure Theory, Integration, and Hilbert Spaces*. Princeton University Press. Cited on pages 147 and 148.
- Strehl, A. L., L. Li, E. Wiewiora, J. Langford, and M. L. Littman
2006. PAC model-free reinforcement learning. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, W. W. Cohen and A. W. Moore, eds., volume 148 of *ACM International Conference Proceeding Series*, Pp. 881–888. ACM. Cited on page 65.
- Sutton, R. S. and A. G. Barto
2018. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book. Cited on pages 16, 56, 61, and 64.
- Sutton, R. S., D. A. McAllester, S. Singh, and Y. Mansour
1999a. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*, S. A. Solla, T. K. Leen, and K. Müller, eds., Pp. 1057–1063. The MIT Press. Cited on pages 71 and 72.
- Sutton, R. S., D. Precup, and S. Singh
1999b. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, 112(1-2):181–211. Cited on page 243.

BIBLIOGRAPHY

- Szepesvári, C.
2010. *Algorithms for Reinforcement Learning*, Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers. Cited on pages 16 and 54.
- Tarski, A.
1955. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309. Cited on page 129.
- Tieleman, T. and G. Hinton
2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 4. Cited on page 56.
- Tolstikhin, I. O., O. Bousquet, S. Gelly, and B. Schölkopf
2018. Wasserstein auto-encoders. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. Cited on pages 180, 183, 201, and 239.
- Tornblom, J. and S. Nadjm-Tehrani
2019. Formal verification of input-output mappings of tree ensembles. *CoRR*, abs/1905.04194. Cited on page 243.
- Towers, M., J. K. Terry, A. Kwiatkowski, J. U. Balis, G. d. Cola, T. Deleu, M. Goulão, A. Kallinteris, A. KG, M. Krimmel, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. J. Tai, A. T. J. Shen, and O. G. Younis
2023. Gymnasium. Cited on page 176.
- Tsitsiklis, J. N.
1994. Asynchronous stochastic approximation and q-learning. *Mach. Learn.*, 16(3):185–202. Cited on pages 16 and 65.
- Tsitsiklis, J. N. and B. V. Roy
1997. An analysis of temporal-difference learning with function approximation. *IEEE Trans. Autom. Control.*, 42(5):674–690. Cited on page 65.
- van den Oord, A., N. Kalchbrenner, L. Espeholt, K. Kavukcuoglu, O. Vinyals, and A. Graves
2016. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, eds., Pp. 4790–4798. Cited on page 239.

- van den Oord, A., O. Vinyals, and K. Kavukcuoglu
2017. Neural discrete representation learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, eds., Pp. 6306–6315. Cited on page 193.
- van der Maaten, L. and G. Hinton
2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605. Cited on pages 238 and 337.
- van der Pol, E., T. Kipf, F. A. Oliehoek, and M. Welling
2020. Plannable approximations to MDP homomorphisms: Equivariance under actions. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20, Auckland, New Zealand, May 9-13, 2020*, A. E. F. Seghrouchni, G. Sukthankar, B. An, and N. Yorke-Smith, eds., Pp. 1431–1439. International Foundation for Autonomous Agents and Multiagent Systems. Cited on page 136.
- van Hasselt, H., A. Guez, and D. Silver
2016. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, D. Schuurmans and M. P. Wellman, eds., Pp. 2094–2100. AAAI Press. Cited on pages 69 and 341.
- Vanderbei, R. J.
1991. Uniform continuity is almost lipschitz continuity. Technical Report SOR-91-11, Statistics and Operations Research Series, Princeton University. Cited on page 231.
- Vardi, M. and P. Wolper
1994. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37. Cited on pages 87 and 88.
- Vaserstein, L. N.
1969. Markov processes over denumerable products of spaces, describing large systems of automata. *Problemy Peredaci Informacii*, 5:64–72. Cited on pages 126 and 127.
- Villani, C.
2009. *Optimal Transport: Old and New*. Berlin, Heidelberg: Springer Berlin Heidelberg. Cited on pages 125, 127, 182, and 186.
- Wang, Y. and X. Tan
2021. Deep recurrent belief propagation network for pomdps. In *Proceedings of the*

- AAAI Conference on Artificial Intelligence, volume 35, Pp. 10236–10244. Cited on page 212.
- Watanabe, K., M. van der Vegt, I. Hasuo, J. Rot, and S. Junges
2024. Pareto curves for compositionally model checking string diagrams of mdps. In TACAS, volume 14571 of LNCS, Pp. 279–298. Springer. Cited on page 242.
- Watkins, C. J. C. H. and P. Dayan
1992. Q-learning. *Mach. Learn.*, 8:279–292. Cited on pages 64 and 65.
- Wells, A. M., M. Lahijanian, L. E. Kavraki, and M. Y. Vardi
2020. Ltlf synthesis on probabilistic systems. In *Proceedings 11th International Symposium on Games, Automata, Logics, and Formal Verification, GandALF 2020, Brussels, Belgium, September 21–22, 2020*, J. Raskin and D. Bresolin, eds., volume 326 of EPTCS, Pp. 166–181. Cited on page 101.
- White, M.
2017. Unifying task specification in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6–11 August 2017*, D. Precup and Y. W. Teh, eds., volume 70 of *Proceedings of Machine Learning Research*, Pp. 3742–3750. PMLR. Cited on page 76.
- Wienhöft, P., M. Suilen, T. D. Simão, C. Dubslaff, C. Baier, and N. Jansen
2023. More for less: Safe policy improvement with stronger performance guarantees. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th–25th August 2023, Macao, SAR, China*, Pp. 4406–4415. ijcai.org. Cited on page 283.
- Wiewiora, E.
2003. Potential-based shaping and q-value initialization are equivalent. *J. Artif. Intell. Res.*, 19:205–208. Cited on page 274.
- Wu, Y., C. Macdonald, and I. Ounis
2021. Partially observable reinforcement learning for dialog-based interactive recommendation. In *Proceedings of the 15th ACM Conference on Recommender Systems*, Pp. 241–251. Cited on page 207.
- Wurman, P. R., S. Barrett, K. Kawamoto, J. MacGlashan, K. Subramanian, T. J. Walsh, R. Capobianco, A. Devlic, F. Eckert, F. Fuchs, L. Gilpin, P. Khandelwal, V. Kompella, H. Lin, P. MacAlpine, D. Oller, T. Seno, C. Sherstan, M. D. Thomure, H. Aghabozorgi, L. Barrett, R. Douglas, D. Whitehead, P. Dürr, P. Stone, M. Spranger, and H. Kitano
2022. Outracing champion gran turismo drivers with deep reinforcement learning. *Nat.*, 602(7896):223–228. Cited on pages 15 and 133.

- Xiao, X., B. Liu, G. Warnell, and P. Stone
2022. Motion planning and control for mobile robot navigation using machine learning: a survey. *Auton. Robots*, 46(5):569–597. Cited on page 15.
- Xiong, Z., I. Agarwal, and S. Jagannathan
2022. HiSaRL: A hierarchical framework for safe reinforcement learning. In *SafeAI*, volume 3087. CEUR-WS.org. Cited on page 244.
- Yang, C., M. Littman, and M. Carbin
2023a. Computably continuous reinforcement-learning objectives are pac-learnable. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, B. Williams, Y. Chen, and J. Neville, eds., Pp. 10729–10736. AAAI Press. Cited on page 111.
- Yang, C., M. L. Littman, and M. Carbin
2022. On the (in)tractability of reinforcement learning for LTL objectives. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, L. D. Raedt, ed., Pp. 3650–3658. ijcai.org. Cited on page 99.
- Yang, W., G. Marra, G. Rens, and L. D. Raedt
2023b. Safe reinforcement learning via probabilistic logic shields. In *IJCAI*, Pp. 5739–5749. ijcai.org. Cited on page 244.
- Young, K. and T. Tian
2019. Minatar: An atari-inspired testbed for more efficient reinforcement learning experiments. *CoRR*, abs/1903.03176. Cited on page 236.
- Zanette, A. and E. Brunskill
2019. Tighter problem-dependent regret bounds in reinforcement learning without domain knowledge using value function bounds. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, K. Chaudhuri and R. Salakhutdinov, eds., volume 97 of *Proceedings of Machine Learning Research*, Pp. 7304–7312. PMLR. Cited on page 57.
- Zang, H., X. Li, and M. Wang
2022. Simsrl: Simple distance-based state representations for deep reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(8):8997–9005. Cited on page 136.

BIBLIOGRAPHY

Zennaro, F. M., P. Turrini, and T. Damoulas

2023. Quantifying consistency and information loss for causal abstraction learning. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, E. Elkind, ed., Pp. 5750–5757. International Joint Conferences on Artificial Intelligence Organization. Main Track. Cited on page 136.

Zhang, A., R. T. McAllister, R. Calandra, Y. Gal, and S. Levine

2021. Learning invariant representations for reinforcement learning without reconstruction. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. Cited on pages 136 and 252.

Zhao, T., H. Hachiya, G. Niu, and M. Sugiyama

2012. Analysis and improvement of policy gradient estimation. *Neural Networks*, 26:118–129. Cited on page 75.

Žikelić, Đ., M. Lechner, A. Verma, K. Chatterjee, and T. A. Henzinger

2023. Compositional policy learning in stochastic control systems with formal guarantees. In *NeurIPS*. Cited on pages 243 and 244.

Åström, K.

1965. Optimal control of markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10(1):174–205. Cited on page 213.

Author's Publications

Avalos, R., F. Delgrange, A. Nowe, G. A. Perez, and D. M. Roijers

2024. The wasserstein believer: Learning belief updates for partially observable environments through reliable latent space models. In *The Twelfth International Conference on Learning Representations*.

Brihaye, T., F. Delgrange, Y. Oualhadj, and M. Randour

2019. Life is random, time is not: Markov decision processes with window objectives. In *30th International Conference on Concurrency Theory, CONCUR 2019, August 27-30, 2019, Amsterdam, the Netherlands*, W. J. Fokkink and R. van Glabbeek, eds., volume 140 of *LIPICs*, Pp. 8:1–8:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Brihaye, T., F. Delgrange, Y. Oualhadj, and M. Randour

2020. Life is random, time is not: Markov decision processes with window objectives. *Logical Methods in Computer Science*, 16(4).

Delgrange, F., G. Avni, A. Lukina, C. Schilling, A. Nowé, and G. A. Pérez

2024. Synthesis of hierarchical controllers based on deep reinforcement learning policies. *CoRR*, abs/2402.13785.

Delgrange, F., J. Katoen, T. Quatmann, and M. Randour

2020. Simple strategies in multi-objective mdps. In *Tools and Algorithms for the Construction and Analysis of Systems - 26th International Conference, TACAS 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25-30, 2020, Proceedings, Part I*, A. Biere and D. Parker, eds., volume 12078 of *Lecture Notes in Computer Science*, Pp. 346–364. Springer.

Delgrange, F., A. Nowé, and G. A. Pérez

2023a. Wasserstein auto-encoded mdps: Formal verification of efficiently distilled RL policies with many-sided guarantees. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Delgrange, F., A. Nowé, and G. A. Pérez

2022. Distillation of rl policies with formal guarantees via variational abstraction of markov decision processes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(6):6497–6505.

Delgrange, F., M. Reymond, A. Nowe, and G. A. Pérez

2023b. Wae-pcn: Wasserstein-autoencoded pareto conditioned networks. In *Proc. of the Adaptive and Learning Agents Workshop (ALA 2023)*, F. Cruz, C. Hayes , C. Wang, and C. Yates, eds., volume <https://alaworkshop2023.github.io/>, Pp. 1–7. 2023 Adaptive and Learning Agents Workshop at AAMAS, ALA 2023 ; Conference date: 29-05-2023 Through 30-05-2023.

Elbarbari, M., F. Delgrange, I. Vervlimmeren, K. Efthymiadis, B. Vanderborght, and A. Nowé

2022. A framework for flexibly guiding learning agents. *Neural Computing and Applications, Special Issue on Adaptive and Learning Agents 2021*.

Appendix

A Details on VAE-MDPs Experiments

A.1 Details on Policy Distillation

The goal of distilling the original RL policy π into a latent policy $\bar{\pi}$ is to find a smaller policy which allows replicating the behaviors of π through the latent spaces. However, executing the lifting of $\bar{\pi}$ in the original model $\mathcal{M}$ may result in performance loss compared to that offered under π . In fact, distillation failure may reveal the lack of robustness of input RL policies, which may produce decisions according to very specific points of the state spaces and provide good performance in a limited set of states. We propose two approaches that can be used alone or together to alleviate this problem.

Robust RL policies. In general, policies π learned through deep RL provide *local* performance, meaning they provide good performance in states $s \in \mathcal{S}$ that are likely to be reached from s_I by executing π . It is, however, often suitable to distill and verify *robust* RL policies, i.e., policies trained to provide good performance in a wider range of states. Training robust RL policies can be achieved by allowing the agent to learn (or pursue its training) in a modified version of $\mathcal{M}$, where s_I is picked at random from $\mathcal{S}$. Nevertheless, training robust deep RL policies this way may take longer time and require more exploration. The degree of randomness can be decided according to the exploration vs. exploitation trade-off and the additional training time allowed. We argue that this additional time is mild in the context of policy verification since enabling and applying model checking are time-consuming tasks by nature.

ε -mimic. When π does not exhibit robustness issues, we allow the encoder to process states reachable under both $\bar{\pi}$ and π by learning from a mixture $\hat{\pi}^\varepsilon$ of the two, defined as $\hat{\pi}^\varepsilon(a | s) = \pi(a | s) \cdot (1 - \varepsilon) + \bar{\pi}(a | s) \cdot \varepsilon$ for all $s \in \mathcal{S}$, $a \in \mathcal{A}$, given $\varepsilon \in [0, 1]$. Assuming π is sufficiently robust, states entered according to a decision $a \sim \bar{\pi}(\cdot | s)$ (produced with probability ε) should be efficiently processed by π . In practice, we may start with a high

value of ε (e.g., $\varepsilon = 1/2$) to encourage exploration of states possibly reachable under $\bar{\pi}$ and we anneal it to 0 during training.

A.2 Setup

We used TENSORFLOW 2.4.1 [Abadi et al., 2015] to implement the neural networks modeling our parameterized distributions constituting our variational model, optimize the loss function (ELBO), and running our PAC approximation schemes. Precisely, we used TENSORFLOW PROBABILITY 0.12.2 [Dillon et al., 2017] to handle the probabilistic components of the VAE (e.g., distributions, reparameterization tricks, etc.), as well as TF-AGENTS 0.7.1 [Guadarrama et al., 2018] to handle the RL parts of the framework, coupled with REVERB replay buffers [Cassirer et al., 2021].

The experiments have been driven on a cluster running under CentOS Linux 7 (Core) composed of a mix of nodes containing Intel processors with the following CPU microarchitectures: (i) 10-core INTEL E5-2680v2, (ii) 14-core INTEL E5-2680v4, and (iii) 20-core INTEL Xeon Gold 6148. We used 8 cores and 128 GB of memory for each of our experiments.

A.3 Hyperparameters

Each distribution parameters (locations, scales, logits) are inferred by feed-forward networks composed of two dense hidden layers and 256 cells by layer. For all our experiments, we used the TENSORFLOW’s implementation of the Adam optimizer [Kingma and Ba, 2015] to learn the neural network weights. We fixed the size of the prioritized experience replay to 10^6 . The training starts when the agent has collected 10^4 transitions in $\mathcal{M}$. Every time a transition is added to the replay buffer, its priority is set to the maximum value. We used minibatches of size 128 to optimize the loss function (i.e., ELBO) and we applied a minibatch update every time the agent executing π performed 16 steps in $\mathcal{M}$. This allows the VAE to process at least $1/8$ of the new transitions collected by sampling from the replay buffer (more details in Schaul et al. 2016). We fixed η_2 to 0.4 as suggested by Schaul et al. [2016].

Annealing schemes. During training, we anneal different hyperparameters ζ to 0 or 1 according to the following annealing schemes: (i) we start annealing the parameters at step $t_0 = 10^4$, (ii) if $\zeta \in [0, \infty]$, then ζ is annealed to 0 via $\zeta_t = \zeta \cdot (1 - \vartheta_\zeta)^{t-t_0}$, or (iii) if $\zeta \in [0, 1]$, then ζ is annealed to 1 via $\zeta_t = \zeta + (1 - \zeta) \cdot (1 - (1 - \vartheta_\zeta)^{t-t_0})$ for step $t > t_0$ and annealing term ϑ_ζ .

Unshared parameters. The list of hyperparameters used for the different environments is given in Table 4. The activation function of all cells of the neural

networks was chosen between relu and leaky relu, the learning rate of the optimizer in $[10^{-4}, 10^{-3}]$, the regularizer scale factor α from $\{0.1, 1, 10, 100\}$, ϑ in $[0, 1]$, the values of each annealing term in $[10^{-6}, 10^{-4}]$, and we tested both loss- and bucket-based experience replay buffers.

A.4 Labeling functions

The labeling functions ℓ we used for each environment are detailed in Table 2.

A.5 Reproducibility of the Results

Seeding. The code we provide allows for setting random seeds to run the experiments, making the overall shape of the Figures we present in this paper reproducible. However, the exact values obtained in each particular instance are not: our prioritized experience replays (and those of TF-AGENTS) rely on REVERB replay buffers, handling randomness via the ABSEIL C++ library¹ which does not allow for manual seeding. All other libraries using randomness are seeded properly. The code for conducting and replicating our experiments is available at https://github.com/florentdelgrange/vae_mdp.

B Details on WAE-MDPs Experiments

B.1 Setup

We used TENSORFLOW 2.7.0 [Abadi et al., 2015] to implement the neural network architecture of our W²AE-MDP, TENSORFLOW PROBABILITY 0.15.0 [Dillon et al., 2017] to handle the probabilistic components of the latent model (e.g., latent distributions with reparameterization tricks, masked autoregressive flows, etc.), as well as TF-AGENTS 0.11.0 [Guadarrama et al., 2018] to handle the RL parts of the framework.

Models have been trained on a cluster running under CentOS Linux 7 (Core) composed of a mix of nodes containing Intel processors with the following CPU microarchitectures: (i) 10-core INTEL E5-2680v2, (ii) 14-core INTEL E5-2680v4, and (iii) 20-core INTEL Xeon Gold 6148. We used 8 cores and 32 GB of memory for each run.

B.2 Hyperparameters

WAE-MDP parameters. All components (e.g., functions or distribution locations and scales, see Figure 3.11) are represented and inferred by feed-forward networks. All the

¹<https://abseil.io>

	CartPole	MountainCar	Acrobot	Pendulum	LunarLander
activation	leaky relu	leaky relu	relu	relu	relu
learning rate	10^{-3}	10^{-3}	10^{-4}	10^{-4}	10^{-4}
RL policy	DQN	DQN	DQN (robust)	SAC (robust)	SAC
$\log_2 \bar{\mathcal{S}} $	9	10	13	13	16
$ \bar{\mathcal{A}} $	/	/	/	3	5
$\lambda_1^{\bar{\mathcal{S}}}$	$2/3$	$2/3$	$2/3$	$2/3$	$2/3$
$\lambda_2^{\bar{\mathcal{S}}}$	$1/2$	$1/2$	$1/2$	$1/2$	$1/2$
$\lambda_1^{\bar{\mathcal{A}}}$	/	/	/	$1/2$	$1/4$
$\lambda_2^{\bar{\mathcal{A}}}$	/	/	/	$1/3$	$1/6$
ϑ_{λ_1}	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}
ϑ_{λ_2}	$2 \cdot 10^{-6}$	$2 \cdot 10^{-6}$	$2 \cdot 10^{-6}$	$2 \cdot 10^{-6}$	$2 \cdot 10^{-6}$
α	10^1	10^1	10^1	10^1	10^1
α_A	/	/	/	1	10^{-1}
ϑ_α	10^{-5}	10^{-5}	$7.5 \cdot 10^{-5}$	$7.5 \cdot 10^{-5}$	10^{-5}
ϑ_β	$5 \cdot 10^{-5}$	$5 \cdot 10^{-5}$	$7.5 \cdot 10^{-5}$	$7.5 \cdot 10^{-5}$	$5 \cdot 10^{-5}$
PER	bucket	bucket	bucket	loss	loss
η_1	$1/3$	$1/3$	$3 \cdot 10^{-1}$	$3 \cdot 10^{-1}$	$3 \cdot 10^{-1}$
ϑ_{η_2}	$7 \cdot 10^{-5}$	$7.5 \cdot 10^{-5}$	$7 \cdot 10^{-5}$	10^{-5}	10^{-5}
ε	0	0	$1/2$	$1/2$	0
ϑ_ε	/	/	10^{-5}	10^{-5}	/

Table 1: Hyperparameter choices for the different environments. We write $\lambda_i^{\mathcal{X}}$ for the temperature parameter used for the concrete relaxation of Bernoulli distributions when $\mathcal{X} = \bar{\mathcal{S}}$ (resp. Categorical distributions when $\mathcal{X} = \bar{\mathcal{A}}$). This corresponds to the encoder distribution if $i = 1$ and the one of the latent transition function (resp. distilled policy) else, if $i = 2$. PER stands for the kind of prioritized experience replay buffer used. Moreover, we write α_A for the scale factor of the action entropy regularizer where the final entropy regularization term is $\alpha \cdot (h(\Phi) + \alpha_A \cdot h(\Phi^A))$.

Environment	$S \subseteq$	Description, for $s \in S$	$\ell(s) = \langle \mathbf{a}_1, \dots, \mathbf{a}_n, \text{reset} \rangle$
CartPole	$\mathbb{R}^4$	s_1: cart position s_2: cart velocity s_3: pole angle (rad) s_4: pole velocity at tip	$\mathbf{a}_1 = \mathbb{1} \{s_1 \geq 1.5\}$: unsafe cart position $\mathbf{a}_2 = \mathbb{1} \{s_3 \geq 0.15\}$: unsafe pole angle
MountainCar	$\mathbb{R}^2$	s_1: position s_2: velocity	$\mathbf{a}_1 = \mathbb{1} \{s_1 > 1.5\}$: target position $\mathbf{a}_2 = \mathbb{1} \{s_1 \geq -1/2\}$: right-hand side of the mountain $\mathbf{a}_3 = \mathbb{1} \{s_2 \geq 0\}$: car going forward
Acrobot	$\mathbb{R}^6$	Let $\theta_1, \theta_2 \in [0, 2\pi]$ be the angles of the two rotational joints, $s_1 = \cos(\theta_1)$ $s_2 = \sin(\theta_1)$ $s_3 = \cos(\theta_2)$ $s_4 = \sin(\theta_2)$ s_5: angular velocity 1 s_6: angular velocity 2	$\mathbf{a}_1 = \mathbb{1} \{-s_1 - s_3 \cdot s_1 + s_4 \cdot s_2 > 1\}$: RL agent target $\mathbf{a}_2 = \mathbb{1} \{s_1 \geq 0\}$: $\theta_1 \in [0, \pi/2] \cup [3\pi/2, 2\pi]$ $\mathbf{a}_3 = \mathbb{1} \{s_2 \geq 0\}$: $\theta_1 \in [0, \pi]$ $\mathbf{a}_4 = \mathbb{1} \{s_3 \geq 0\}$: $\theta_2 \in [0, \pi/2] \cup [3\pi/2, 2\pi]$ $\mathbf{a}_5 = \mathbb{1} \{s_4 \geq 0\}$: $\theta_2 \in [0, \pi]$ $\mathbf{a}_6 = \mathbb{1} \{s_5 \geq 0\}$: positive angular velocity (1) $\mathbf{a}_7 = \mathbb{1} \{s_6 \geq 0\}$: positive angular velocity (2)
Pendulum	$\mathbb{R}^3$	Let $\theta \in [0, 2\pi]$ be the joint angle $s_1 = \cos(\theta)$ $s_2 = \sin(\theta)$ s_3: angular velocity	$\mathbf{a}_1 = \mathbb{1} \{s_1 \geq \cos(\pi/3)\}$: safe joint angle $\mathbf{a}_2 = \mathbb{1} \{s_1 \geq 0\}$: $\theta \in [0, \pi/2] \cup [3\pi/2, 2\pi]$ $\mathbf{a}_3 = \mathbb{1} \{s_2 \geq 0\}$: $\theta \in [0, \pi]$ $\mathbf{a}_4 = \mathbb{1} \{s_3 \geq 0\}$: positive angular velocity
LunarLander	$\mathbb{R}^8$	s_1: horizontal coordinates s_2: vertical coordinates s_3: horizontal speed s_4: vertical speed s_5: ship angle s_6: angular speed s_7: left leg contact s_8: right leg contact	$\mathbf{a}_1$: unsafe angle $\mathbf{a}_2$: leg ground contact $\mathbf{a}_3$: lands rapidly $\mathbf{a}_4$: left inclination $\mathbf{a}_5$: right inclination $\mathbf{a}_6$: motors shut down

Table 2: Labeling functions for the OPENAI gym environments considered in our experiments. We provide a short description of the state space and the meaning of each atomic proposition. Recall that labels are binary encoded, for $n = |\text{AP}| - 1$ (one bit is reserved for `reset`, cf. Assumption 5).

networks share the same architecture (i.e., number of layers and neurons per layer). We use a simple uniform experience replay of size 10^6 to store the transitions and sample them. The training starts when the agent has collected 10^4 transitions in $\mathcal{M}$. We used minibatches of size 128 to optimize the objective and we applied a minibatch update every time the agent executing π has performed 16 steps in $\mathcal{M}$. We use the recursive ϵ -perturbation trick of Huang [2020] with $\epsilon = 3/4$: when an episode ends, it restarts from the initial state with probability $1/4$; before re-starting an episode, the time spent in the reset state labeled with **reset** follows then the geometric distribution with expectation $\epsilon/1-\epsilon = 3$. We chose the same latent state-action space size as for VAE-MDPs, except for LunarLander that we decreased to $\log_2 |\bar{S}| = 14$ and $|\bar{A}| = 3$ to improve the scalability of the verification.

VAE-MDPs parameters. For the comparison of Section 3.6, we used the exact same VAE-MDP hyperparameter set as prescribed in Appendix A, except for the state-action space of LunarLander that we also changed for scalability and fair comparison purpose.

Hyperparameter search. To evaluate our WAE-MDP, we realized a search in the parameter space defined in Table 3. The best parameters found (in terms of trade-off between performance and latent quality) are reported in Table 4. We used two different optimizers for minimizing the loss (referred to as the minimizer) and computing the Wasserstein terms (referred to as the maximizer). We used ADAM [Kingma and Ba, 2015] for the two, but we allow for different learning rates ADAM_α and exponential decays $\text{ADAM}_{\beta_1}, \text{ADAM}_{\beta_2}$. We also found that polynomial decay for ADAM_α (e.g., to 10^{-5} for $4 \cdot 10^5$ steps) is a good practice to stabilize the experiment learning curves, but is not necessary to obtain high-quality and performing distillation. Concerning the continuous relaxation of discrete distributions, we used a different temperature for each distribution, as Maddison et al. [2017] pointed out that doing so is valuable to improve the results. We further followed the guidelines of Maddison et al. [2017] to choose the interval of temperatures and did not schedule any annealing scheme (in contrast to VAE-MDPs). Essentially, the search reveals that the regularizer scale factors β (defining the optimization direction) as well as the encoder and latent transition temperatures are important to improve the performance of distilled policies. For the encoder temperature, we found a nice spot in $\lambda_\phi = 2/3$, which provides the best performance in general, whereas the choice of $\lambda_{\bar{\phi}}$ and β are (latent-) environment dependent. The importance of the temperature parameters for the continuous relaxation of discrete distributions is consistent with the results of [Maddison et al., 2017], revealing that the success of the relaxation depends on the choice of the temperature for the different latent space sizes.

B.3 Reproducibility of the Results

Code. The code for conducting and replicating our experiments is available at https://github.com/florentdelgrange/wae_mdp.

Table 3: Hyperparameter search. λ_X refers to the temperature used for W²AE-MDP component X .

Parameter	Range
ADAM_α (minimizer)	$\{0.0001, 0.0002, 0.0003, 0.001\}$
ADAM_α (maximizer)	$\{0.0001, 0.0002, 0.0003, 0.001\}$
ADAM_{β_1}	$\{0, 0.5, 0.9\}$
ADAM_{β_2}	$\{0.9, 0.999\}$
neurons per layer	$\{64, 128, 256, 512\}$
number of hidden layers	$\{1, 2, 3\}$
activation	$\left\{ \text{ReLU}, \text{Leaky ReLU}, \tanh, \frac{\text{softplus}(2x+2)}{2} - 1 \text{ (smooth ELU)} \right\}$
$\beta_{L_{\xi\pi}}$	$\{10, 25, 50, 75, 100\}$
β_{L_P}	$\{10, 25, 50, 75, 100\}$
n_{reg}	$\{5, 10, 15, 20\}$
δ	$\{10, 20\}$
use ε -mimic	$\{\text{True}, \text{False}\}$ (if True, a decay rate of 10^{-5} is used)
$\lambda_{\bar{P}}$	$\{0.1, 1/3, 1/2, 2/3, 3/5, 0.99\}$
λ_ϕ	$\{0.1, 1/3, 1/2, 2/3, 3/5, 0.99\}$
$\lambda_{\bar{\pi}}$	$\left\{ 1/ \bar{A} -1, 1/(\bar{A} -1)^{1.5} \right\}$
λ_{ϕ^A}	$\left\{ 1/ \bar{A} -1, 1/(\bar{A} -1)^{1.5} \right\}$

Table 4: Final hyperparameters used to evaluate WAE-MDPs in Section 3.6

	CartPole	MountainCar	Acrobot	LunarLander	Pendulum
$\log_2 \bar{\mathcal{S}} $	9	10	13	14	13
$ \bar{\mathcal{A}} $	$2 = \mathcal{A} $	$2 = \mathcal{A} $	$3 = \mathcal{A} $	3	3
activation	tanh	ReLU	Leaky Relu	ReLU	ReLU
layers	[64, 64, 64]	[512, 512]	[512, 512]	[256]	[256, 256, 256]
ADAM_{α} (minimizer)	0.0002	0.0001	0.0002	0.0003	0.0003
ADAM_{α} (maximizer)	0.0002	0.0001	0.0001	0.0003	0.0003
ADAM_{β_1}	0.5	0	0	0	0.5
ADAM_{β_2}	0.999	0.999	0.999	0.999	0.999
β_{L_P}	10	25	10	50	25
$\beta_{L^{\xi_{\pi}}}$	75	100	10	100	25
n_{reg}	5	20	20	15	5
δ	20	10	20	20	10
ε	0	0	0	0	0.5
$\lambda_{\bar{\mathbf{p}}}$	$1/3$	$1/3$	0.1	0.75	$2/3$
$\lambda_{\bar{\phi}}$	$1/3$	$2/3$	$2/3$	$2/3$	$2/3$
$\lambda_{\bar{\pi}}$	$2/3$	$1/3$	0.5	0.5	0.5
$\lambda_{\bar{\phi}^{\mathcal{A}}}$	/	/	/	$1/3$	$1/3$

C The Wasserstein Believer: Supplementary Material

C.1 Proof of Theorem 30

Warm up: Some Wasserstein Properties

In the following, we elaborate on properties and definitions related to the Wasserstein metrics that will be useful to prove the main claims.

Lemma 25: IPM constant

Let $f : \mathcal{X} \rightarrow \mathbb{R}$ and d be a metric on $\mathcal{X}$. Denote by

$$\text{Lipsch}_d^K = \{f : f(x_1) - f(x_2) \leq K \cdot d(x_1, x_2)\}$$

the set of K -Lipschitz functions with respect to d . Assume that f is K -Lipschitz, i.e., $f \in \text{Lipsch}_d^K$, then for any two distributions $P, Q \in \Delta(\mathcal{X})$,

$$\left| \mathbb{E}_{x_1 \sim P} f(x_1) - \mathbb{E}_{x_2 \sim Q} f(x_2) \right| \leq K \cdot \mathcal{W}_d(P, Q).$$

In the case where $d = 1_\neq$, for **any** bounded function $g : \mathcal{X} \rightarrow Y$ with $Y \subseteq \mathbb{R}$, we have $|\mathbb{E}_{x_1 \sim P} g(x_1) - \mathbb{E}_{x_2 \sim Q} g(x_2)| \leq K_Y \cdot \mathcal{W}_{1_\neq}(P, Q) = K_Y \cdot d_{TV}(P, Q)$, where $K_Y \geq \sup_{x \in \mathcal{X}} |g(x)|$.

In both cases, we call the smallest K (resp. K_Y) the *integral probability metric (IPM, Müller, 1997; Sriperumbudur et al., 2009) constant* of the function f (resp. g).

Proof. Let $f \in \text{Lipsch}_d^K$, and $P, Q \in \Delta(\mathcal{X})$. Take $f' = f/K \in \text{Lipsch}_d^1$. Then, we have that $|\mathbb{E}_P f - \mathbb{E}_Q f| = K \cdot |\mathbb{E}_P f' - \mathbb{E}_Q f'| \leq K \cdot \mathcal{W}_d(P, Q)$. The reasoning is the same for bounded functions g : let $K_Y \geq \sup_{x \in \mathcal{X}} |g(x)|$ and $g' = g/K_Y$. It is clear that g' is one-bounded, i.e., $\|g'\|_\infty = \sup_{x \in \mathcal{X}} |g'(x)| \leq 1$. Recall that d_{TV} can be formalized as an integral probability metric [Müller, 1997], with $d_{TV}(P, Q) = \sup_{\|f\|_\infty \leq 1} |\mathbb{E}_P f - \mathbb{E}_Q f|$ (Definition 34). Then, we have that $|\mathbb{E}_P g - \mathbb{E}_Q g| = K_Y \cdot |\mathbb{E}_P g' - \mathbb{E}_Q g'| \leq K_Y \cdot d_{TV}(P, Q) = K_Y \cdot \mathcal{W}_{1_\neq}(P, Q)$. $\square$

Lemma 25 intuitively explains the occurrence of the constant $K_{\bar{V}}$ in the bound of Theorem 30: since the value function is bounded by $K_{\bar{V}}$, given two distributions over the discrete space $\bar{\mathcal{S}}$, the absolute difference in the value functions of the belief states is bounded by $K_{\bar{V}}$ times the Wasserstein distance between the two distributions.

Since the value difference is computed in expectation (over histories drawn from $\mathcal{H}_{\bar{\pi}}$), we introduce the following useful property of the Wasserstein distance:

Lemma 26: Wasserstein in expectation

For any function $f : \mathcal{Y} \times \mathcal{X} \rightarrow \mathbb{R}$ so that $\mathcal{X}$ is equipped with the metric d , consider the function $g_y : \mathcal{X} \rightarrow \mathbb{R}$ defined as $g_y(x) = f(y, x)$. Assume that for any $y \in \mathcal{Y}$, g_y is Lipschitz (or simply bounded if $d = \mathbb{1}_{\neq}$) and let K be the IPM constant of g_y . Then, let $\mathcal{D} \in \Delta(\mathcal{Y})$ be a distribution over $\mathcal{Y}$ and $P, Q \in \Delta(\mathcal{X})$ be two distributions over $\mathcal{X}$, we have $\mathbb{E}_{y \sim \mathcal{D}} |\mathbb{E}_{x_1 \sim P} f(y, x_1) - \mathbb{E}_{x_2 \sim Q} f(y, x_2)| \leq K \cdot \mathcal{W}_d(P, Q)$.

Proof. The proof is straightforward by construction of g_y :

$$\begin{aligned} & \mathbb{E}_{y \sim \mathcal{D}} \left| \mathbb{E}_{x_1 \sim P} f(y, x_1) - \mathbb{E}_{x_2 \sim Q} f(y, x_2) \right| \\ &= \mathbb{E}_{y \sim \mathcal{D}} \left| \mathbb{E}_{x_1 \sim P} g_y(x_1) - \mathbb{E}_{x_2 \sim Q} g_y(x_2) \right| \\ &\leq \mathbb{E}_{y \sim \mathcal{D}} [K \cdot \mathcal{W}_d(P, Q)] \quad (\text{Lemma 25}) \\ &= K \cdot \mathcal{W}_d(P, Q) \end{aligned}$$

□

Model Quality Bound: Time to Raise your Expectations

Let us restate the theorem:

Theorem 30: Model quality

For any latent policy $\bar{\pi} : \bar{\mathcal{B}} \rightarrow \Delta(\mathcal{A})$, the values of $\mathcal{P}$ and $\bar{\mathcal{P}}$ are bounded by the local and belief losses in average when, in both models, the actions are produced by $\bar{\pi}$, which is conditioned on the latent belief induced by Φ , i.e., $a \sim \bar{\pi}(\cdot \mid \Phi^*(h))$:

$$\mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} |V^{\bar{\pi}}(h) - \bar{V}^{\bar{\pi}}(h)| \leq \frac{\mathcal{L}}{1 - \gamma}. \quad (4.7)$$

Proof. The absolute value difference can be bounded by:

$$\begin{aligned} & \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} |V^{\bar{\pi}}(h) - \bar{V}^{\bar{\pi}}(h)| \\ &= \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{a \sim \bar{\pi}(\cdot \mid \Phi^*(h))} \left[\mathcal{R}(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(\cdot \mid s, a)} \mathbb{E}_{o' \sim \mathcal{O}(\cdot \mid s', a)} V^{\bar{\pi}}(h \cdot a \cdot o') \right] \right. \\ & \quad \left. - \mathbb{E}_{\bar{s} \sim \bar{\mathcal{T}}^*(h)} \mathbb{E}_{a \sim \bar{\pi}(\cdot \mid \Phi^*(h))} \left[\bar{\mathcal{R}}(\bar{s}, a) + \gamma \mathbb{E}_{\bar{s}' \sim \bar{\mathcal{P}}(\cdot \mid \bar{s}, a)} \mathbb{E}_{o' \sim \bar{\mathcal{O}}(\cdot \mid \bar{s}', a)} \bar{V}^{\bar{\pi}}(h \cdot a \cdot o') \right] \right| \end{aligned}$$

$$\begin{aligned}
 &= \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \left[\mathcal{R}(s, a) + \gamma \mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, a)} \mathbb{E}_{o' \sim \mathcal{O}(\cdot | s', a)} V^{\bar{\pi}}(h \cdot a \cdot o') \right] \right. \\
 &\quad \left. - \mathbb{E}_{\bar{s} \sim \bar{\mathcal{T}}^*(h)} \left[\bar{\mathcal{R}}(\bar{s}, a) + \gamma \mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot | \bar{s}, a)} \mathbb{E}_{o' \sim \bar{\mathcal{O}}(\cdot | \bar{s}', a)} \bar{V}^{\bar{\pi}}(h \cdot a \cdot o') \right] \right| \\
 &\hspace{15em} \text{(Jensen's inequality)} \\
 &\leq \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left[\left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathcal{R}(s, a) - \mathbb{E}_{\bar{s} \sim \bar{\mathcal{T}}^*(h)} \bar{\mathcal{R}}(\bar{s}, a) \right| \right. \\
 &\quad \left. + \gamma \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, a)} \mathbb{E}_{o' \sim \mathcal{O}(\cdot | s', a)} V^{\pi}(h \cdot a \cdot o') - \mathbb{E}_{\bar{s} \sim \bar{\mathcal{T}}^*(h)} \mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot | \bar{s}, a)} \mathbb{E}_{o' \sim \bar{\mathcal{O}}(\cdot | \bar{s}', a)} \bar{V}^{\pi}(h \cdot a \cdot o') \right| \right] \\
 &\hspace{15em} \text{(Triangle inequality)}
 \end{aligned}$$

For the sake of clarity, we split the inequality into two parts.

Part 1: Reward bounds

$$\begin{aligned}
 &\mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathcal{R}(s, a) - \mathbb{E}_{\bar{s} \sim \bar{\mathcal{T}}^*(h)} \bar{\mathcal{R}}(\bar{s}, a) \right| \\
 &= \mathbb{E}_{h, o \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \left[\mathcal{R}(s, a) - \bar{\mathcal{R}}(\phi(s, o), a) \right] \right. \\
 &\quad \left. + \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{\bar{s} \sim \bar{\mathcal{T}}^*(h)} \left[\bar{\mathcal{R}}(\phi(s, o), a) - \bar{\mathcal{R}}(\bar{s}, a) \right] \right| \\
 &\hspace{10em} (o \text{ is the last observation of } h; \phi \text{ comes into play}) \\
 &\leq \mathbb{E}_{h, o \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left[\left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \left[\mathcal{R}(s, a) - \bar{\mathcal{R}}(\phi(s, o), a) \right] \right| \right. \\
 &\quad \left. + \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{\bar{s} \sim \bar{\mathcal{T}}^*(h)} \left[\bar{\mathcal{R}}(\phi(s, o), a) - \bar{\mathcal{R}}(\bar{s}, a) \right] \right| \right] \\
 &\hspace{15em} \text{(Triangle inequality)} \\
 &\leq \mathbb{E}_{h, o \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left[\mathbb{E}_{s \sim \mathcal{T}^*(h)} \left| \mathcal{R}(s, a) - \bar{\mathcal{R}}(\phi(s, o), a) \right| \right. \\
 &\quad \left. + \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{\bar{s} \sim \bar{\mathcal{T}}^*(h)} \left[\bar{\mathcal{R}}(\phi(s, o), a) - \bar{\mathcal{R}}(\bar{s}, a) \right] \right| \right] \\
 &\hspace{15em} \text{(Jensen's inequality)} \\
 &= \mathbb{E}_{h, o \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \mathbb{E}_{s \sim \mathcal{T}^*(h)} \left| \mathcal{R}(s, a) - \bar{\mathcal{R}}(\phi(s, o), a) \right| \\
 &\quad + \mathbb{E}_{h, o \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{\bar{s} \sim \bar{\mathcal{T}}^*(h)} \left[\bar{\mathcal{R}}(\phi(s, o), a) - \bar{\mathcal{R}}(\bar{s}, a) \right] \right|
 \end{aligned}$$

$$\begin{aligned}
&= L_{\mathcal{R}} + \mathbb{E}_{h, o \sim H_{\pi}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{\bar{s} \sim \bar{\mathcal{T}}^*(h)} [\bar{\mathcal{R}}(\phi(s, o), a) - \bar{\mathcal{R}}(\bar{s}, a)] \right| \quad (\text{by definition of } L_{\mathcal{R}}) \\
&= L_{\mathcal{R}} + \mathbb{E}_{h, o \sim H_{\pi}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{\bar{s} \sim \bar{\mathcal{T}}^*(h)} \mathbb{E}_{\bar{s}_{\perp} \sim \Phi^*(h)} \left[[\bar{\mathcal{R}}(\phi(s, o), a) - \bar{\mathcal{R}}(\bar{s}_{\perp}, a)] \right. \right. \\
&\quad \left. \left. + [\bar{\mathcal{R}}(\bar{s}_{\perp}, a) - \bar{\mathcal{R}}(\bar{s}, a)] \right] \right| \quad (\text{the belief encoder } \Phi \text{ comes into play}) \\
&= L_{\mathcal{R}} + \mathbb{E}_{h, o \sim H_{\pi}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{\bar{s} \sim \Phi^*(h)} [\bar{\mathcal{R}}(\phi(s, o), a) - \bar{\mathcal{R}}(\bar{s}, a)] \right. \\
&\quad \left. + \mathbb{E}_{\bar{s} \sim \bar{\mathcal{T}}^*(h)} \mathbb{E}_{\bar{s}_{\perp} \sim \Phi^*(h)} [\bar{\mathcal{R}}(\bar{s}_{\perp}, a) - \bar{\mathcal{R}}(\bar{s}, a)] \right| \\
&\leq L_{\mathcal{R}} + \mathbb{E}_{h, o \sim H_{\pi}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left[\left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{\bar{s} \sim \Phi^*(h)} [\bar{\mathcal{R}}(\phi(s, o), a) - \bar{\mathcal{R}}(\bar{s}, a)] \right| \right. \\
&\quad \left. + \left| \mathbb{E}_{\bar{s} \sim \bar{\mathcal{T}}^*(h)} \mathbb{E}_{\bar{s}_{\perp} \sim \Phi^*(h)} [\bar{\mathcal{R}}(\bar{s}_{\perp}, a) - \bar{\mathcal{R}}(\bar{s}, a)] \right| \right] \quad (\text{Triangle inequality}) \\
&\leq L_{\mathcal{R}} + \mathbb{E}_{h, o \sim H_{\pi}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left[\mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{\bar{s} \sim \Phi^*(h)} |\bar{\mathcal{R}}(\phi(s, o), a) - \bar{\mathcal{R}}(\bar{s}, a)| \right. \\
&\quad \left. + \left| \mathbb{E}_{\bar{s} \sim \bar{\mathcal{T}}^*(h)} \mathbb{E}_{\bar{s}_{\perp} \sim \Phi^*(h)} [\bar{\mathcal{R}}(\bar{s}_{\perp}, a) - \bar{\mathcal{R}}(\bar{s}, a)] \right| \right] \quad (\text{Jensen's inequality}) \\
&= L_{\mathcal{R}} + \mathbb{E}_{h, o \sim H_{\pi}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{\bar{s} \sim \Phi^*(h)} |\bar{\mathcal{R}}(\phi(s, o), a) - \bar{\mathcal{R}}(\bar{s}, a)| \\
&\quad + \mathbb{E}_{h, o \sim H_{\pi}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{\bar{s} \sim \bar{\mathcal{T}}^*(h)} \mathbb{E}_{\bar{s}_{\perp} \sim \Phi^*(h)} \bar{\mathcal{R}}(\bar{s}_{\perp}, a) - \bar{\mathcal{R}}(\bar{s}, a) \right| \\
&= L_{\mathcal{R}} + L_{\bar{\mathcal{R}}}^{\Phi} + \mathbb{E}_{h, o \sim H_{\pi}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{\bar{s} \sim \bar{\mathcal{T}}^*(h)} \mathbb{E}_{\bar{s}_{\perp} \sim \Phi^*(h)} [\bar{\mathcal{R}}(\bar{s}_{\perp}, a) - \bar{\mathcal{R}}(\bar{s}, a)] \right| \\
&\quad (\text{by definition of } L_{\bar{\mathcal{R}}}^{\Phi}, \text{ Equation 4.6}) \\
&\leq L_{\mathcal{R}} + L_{\bar{\mathcal{R}}}^{\Phi} + \mathbb{E}_{h \sim H_{\pi}} R_{\text{MAX}} \mathcal{W}_{\bar{d}}(\bar{\mathcal{T}}^*(h), \Phi^*(h)) \quad (\text{by Lemmae 25 and 26}) \\
&= L_{\mathcal{R}} + L_{\bar{\mathcal{R}}}^{\Phi} + R_{\text{MAX}} L_{\bar{\mathcal{T}}};
\end{aligned}$$

Part 2: Next value bounds

$$\begin{aligned}
 & \gamma \cdot \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, a)} \mathbb{E}_{o' \sim \mathcal{O}(\cdot | s', a)} V^{\bar{\pi}}(h \cdot a \cdot o') \right. \\
 & \quad \left. - \mathbb{E}_{\bar{s} \sim \bar{\mathcal{T}}^*(h)} \mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot | \bar{s}, a)} \mathbb{E}_{o' \sim \bar{\mathcal{O}}(\cdot | \bar{s}')} \bar{V}^{\bar{\pi}}(h \cdot a \cdot o') \right| \\
 &= \gamma \cdot \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, a)} \mathbb{E}_{o' \sim \mathcal{O}(\cdot | s', a)} \left[V^{\bar{\pi}}(h \cdot a \cdot o') - \mathbb{E}_{\hat{o}' \sim \bar{\mathcal{O}}(\cdot | \phi(s', o'))} \bar{V}^{\bar{\pi}}(h \cdot a \cdot \hat{o}') \right] \right. \\
 & \quad + \left[\mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, a)} \mathbb{E}_{o' \sim \mathcal{O}(\cdot | s', a)} \mathbb{E}_{\hat{o}' \sim \bar{\mathcal{O}}(\cdot | \phi(s', o'))} \bar{V}^{\bar{\pi}}(h \cdot a \cdot \hat{o}') \right. \\
 & \quad \left. \left. - \mathbb{E}_{\bar{s} \sim \bar{\mathcal{T}}^*(h)} \mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot | \bar{s}, a)} \mathbb{E}_{o' \sim \bar{\mathcal{O}}(\cdot | \bar{s}')} \bar{V}^{\bar{\pi}}(h \cdot a \cdot o') \right] \right| \\
 & \quad (\phi \text{ comes into play, as well as } \bar{\mathcal{O}}) \\
 &\leq \gamma \cdot \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, a)} \mathbb{E}_{o' \sim \mathcal{O}(\cdot | s', a)} \left[V^{\bar{\pi}}(h \cdot a \cdot o') - \mathbb{E}_{\hat{o}' \sim \bar{\mathcal{O}}(\cdot | \phi(s', o'))} \bar{V}^{\bar{\pi}}(h \cdot a \cdot \hat{o}') \right] \right| \\
 & \quad + \gamma \cdot \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, a)} \mathbb{E}_{o' \sim \mathcal{O}(\cdot | s', a)} \mathbb{E}_{\hat{o}' \sim \bar{\mathcal{O}}(\cdot | \phi(s', o'))} \bar{V}^{\bar{\pi}}(h \cdot a \cdot \hat{o}') \right. \\
 & \quad \left. - \mathbb{E}_{\bar{s} \sim \bar{\mathcal{T}}^*(h)} \mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot | \bar{s}, a)} \mathbb{E}_{o' \sim \bar{\mathcal{O}}(\cdot | \bar{s}')} \bar{V}^{\bar{\pi}}(h \cdot a \cdot o') \right| \\
 & \quad (\text{Triangle inequality}) \\
 &= \gamma \cdot \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, a)} \mathbb{E}_{o' \sim \mathcal{O}(\cdot | s', a)} \left[V^{\bar{\pi}}(h \cdot a \cdot o') - \mathbb{E}_{\hat{o}' \sim \bar{\mathcal{O}}(\cdot | \phi(s', o'))} \bar{V}^{\bar{\pi}}(h \cdot a \cdot \hat{o}') \right] \right| \\
 & \quad + \gamma \cdot \mathbb{E}_{h, o \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \left[\mathbb{E}_{s', o' \sim \mathbf{P}_{\Omega}(\cdot | s, o, a)} \mathbb{E}_{\hat{o}' \sim \bar{\mathcal{O}}(\cdot | \phi(s', o'))} \bar{V}^{\bar{\pi}}(h \cdot a \cdot \hat{o}') \right. \right. \\
 & \quad \left. \left. - \mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot | \phi(s, o), a)} \mathbb{E}_{\hat{o}' \sim \bar{\mathcal{O}}(\cdot | \bar{s}')} \bar{V}^{\bar{\pi}}(h \cdot a \cdot \hat{o}') \right] \right. \\
 & \quad + \left[\mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot | \phi(s, o), a)} \mathbb{E}_{o' \sim \bar{\mathcal{O}}(\cdot | \bar{s}')} \bar{V}^{\bar{\pi}}(h \cdot a \cdot o') \right. \\
 & \quad \left. \left. - \mathbb{E}_{\bar{s} \sim \bar{\mathcal{T}}^*(h)} \mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot | \bar{s}, a)} \mathbb{E}_{o' \sim \bar{\mathcal{O}}(\cdot | \bar{s}')} \bar{V}^{\bar{\pi}}(h \cdot a \cdot o') \right] \right| \\
 & \quad (o \text{ is the last observation of } h; \bar{\mathbf{P}} \text{ comes into play}) \\
 &\leq \gamma \cdot \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, a)} \mathbb{E}_{o' \sim \mathcal{O}(\cdot | s', a)} \left[V^{\bar{\pi}}(h \cdot a \cdot o') - \mathbb{E}_{\hat{o}' \sim \bar{\mathcal{O}}(\cdot | \phi(s', o'))} \bar{V}^{\bar{\pi}}(h \cdot a \cdot \hat{o}') \right] \right|
 \end{aligned}$$

$$\begin{aligned}
& + \gamma \cdot \mathbb{E}_{h,o \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \left[\mathbb{E}_{s', o' \sim \mathcal{P}_{\Omega}(\cdot | s, o, a)} \mathbb{E}_{\hat{o}' \sim \bar{\mathcal{O}}(\cdot | \phi(s', o'))} \bar{V}^{\bar{\pi}}(h \cdot a \cdot \hat{o}') \right. \right. \\
& \quad \left. \left. - \mathbb{E}_{\bar{s}' \sim \bar{\mathcal{P}}(\cdot | \phi(s, o), a)} \mathbb{E}_{\hat{o}' \sim \bar{\mathcal{O}}(\cdot | \bar{s}')} \bar{V}^{\bar{\pi}}(h \cdot a \cdot \hat{o}') \right] \right| \\
& + \gamma \cdot \mathbb{E}_{h,o \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{\bar{s}' \sim \bar{\mathcal{P}}(\cdot | \phi(s, o), a)} \mathbb{E}_{o' \sim \bar{\mathcal{O}}(\cdot | \bar{s}')} \bar{V}^{\bar{\pi}}(h \cdot a \cdot o') \right. \\
& \quad \left. - \mathbb{E}_{\bar{s} \sim \bar{\mathcal{T}}^*(h)} \mathbb{E}_{\bar{s}' \sim \bar{\mathcal{P}}(\cdot | \bar{s}, a)} \mathbb{E}_{o' \sim \bar{\mathcal{O}}(\cdot | \bar{s}')} \bar{V}^{\bar{\pi}}(h \cdot a \cdot o') \right| \\
& \hspace{15em} \text{(Triangle inequality)} \\
& \leq \gamma \cdot \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{s' \sim \mathcal{P}(\cdot | s, a)} \mathbb{E}_{o' \sim \bar{\mathcal{O}}(\cdot | s', a)} \left[V^{\bar{\pi}}(h \cdot a \cdot o') - \mathbb{E}_{\hat{o}' \sim \bar{\mathcal{O}}(\cdot | \phi(s', o'))} \bar{V}^{\bar{\pi}}(h \cdot a \cdot \hat{o}') \right] \right| \\
& + \gamma \cdot \mathbb{E}_{h,o \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \mathbb{E}_{s \sim \mathcal{T}^*(h)} \left| \mathbb{E}_{\bar{s}' \sim \phi \mathcal{P}_{\Omega}(\cdot | s, o, a)} \mathbb{E}_{o' \sim \bar{\mathcal{O}}(\cdot | \bar{s}')} \bar{V}^{\bar{\pi}}(h \cdot a \cdot o') \right. \\
& \quad \left. - \mathbb{E}_{\bar{s}' \sim \bar{\mathcal{P}}(\cdot | \phi(s, o), a)} \mathbb{E}_{o' \sim \bar{\mathcal{O}}(\cdot | \bar{s}')} \bar{V}^{\bar{\pi}}(h \cdot a \cdot o') \right| \\
& + \gamma \cdot \mathbb{E}_{h,o \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{\bar{s}' \sim \bar{\mathcal{P}}(\cdot | \phi(s, o), a)} \mathbb{E}_{o' \sim \bar{\mathcal{O}}(\cdot | \bar{s}')} \bar{V}^{\bar{\pi}}(h \cdot a \cdot o') \right. \\
& \quad \left. - \mathbb{E}_{\bar{s} \sim \bar{\mathcal{T}}^*(h)} \mathbb{E}_{\bar{s}' \sim \bar{\mathcal{P}}(\cdot | \bar{s}, a)} \mathbb{E}_{o' \sim \bar{\mathcal{O}}(\cdot | \bar{s}')} \bar{V}^{\bar{\pi}}(h \cdot a \cdot o') \right| \\
& \hspace{15em} \text{(by definition of } \phi \mathcal{P}_{\Omega} \text{ and the Jensen's inequality)} \\
& \leq \gamma \cdot \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{s' \sim \mathcal{P}(\cdot | s, a)} \mathbb{E}_{o' \sim \bar{\mathcal{O}}(\cdot | s', a)} \left[V^{\bar{\pi}}(h \cdot a \cdot o') - \mathbb{E}_{\hat{o}' \sim \bar{\mathcal{O}}(\cdot | \phi(s', o'))} \bar{V}^{\bar{\pi}}(h \cdot a \cdot \hat{o}') \right] \right| \\
& + \gamma \cdot \mathbb{E}_{h,o \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \mathbb{E}_{s \sim \mathcal{T}^*(h)} K_{\bar{\gamma}} \cdot \mathcal{W}_{\bar{d}}(\phi \mathcal{P}_{\Omega}(\cdot | s, a), \bar{\mathcal{P}}(\cdot | \phi(s, o), a)) \\
& + \gamma \cdot \mathbb{E}_{h,o \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{\bar{s}' \sim \bar{\mathcal{P}}(\cdot | \phi(s, o), a)} \mathbb{E}_{o' \sim \bar{\mathcal{O}}(\cdot | \bar{s}')} \bar{V}^{\bar{\pi}}(h \cdot a \cdot o') \right. \\
& \quad \left. - \mathbb{E}_{\bar{s} \sim \bar{\mathcal{T}}^*(h)} \mathbb{E}_{\bar{s}' \sim \bar{\mathcal{P}}(\cdot | \bar{s}, a)} \mathbb{E}_{o' \sim \bar{\mathcal{O}}(\cdot | \bar{s}')} \bar{V}^{\bar{\pi}}(h \cdot a \cdot o') \right| \\
& \hspace{15em} \text{(by Lemma 26)}
\end{aligned}$$

$$\begin{aligned}
&\leq \gamma \cdot \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, a)} \mathbb{E}_{o' \sim \mathcal{O}(\cdot | s', a)} \left[V^{\bar{\pi}}(h \cdot a \cdot o') - \mathbb{E}_{\hat{o}' \sim \bar{\mathcal{O}}(\cdot | \phi(s', o'))} \bar{V}^{\bar{\pi}}(h \cdot a \cdot \hat{o}') \right] \right| \\
&\quad + \gamma K_{\bar{V}} L_{\mathbf{P}} \\
&\quad + \gamma \cdot \mathbb{E}_{h, o \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{\bar{s} \sim \Phi^*(h)} \left| \mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot | \phi(s, o), a)} \left[\mathbb{E}_{o' \sim \bar{\mathcal{O}}(\cdot | \bar{s}')} \bar{V}^{\bar{\pi}}(h \cdot a \cdot o') \right] \right. \\
&\quad \quad \quad \left. - \mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot | \bar{s}, a)} \left[\mathbb{E}_{o' \sim \bar{\mathcal{O}}(\cdot | \bar{s}')} \bar{V}^{\bar{\pi}}(h \cdot a \cdot o') \right] \right| \\
&\quad + \gamma \cdot \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{\bar{s} \sim \Phi^*(h)} \mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot | \bar{s}, a)} \mathbb{E}_{o' \sim \bar{\mathcal{O}}(\cdot | \bar{s}')} \bar{V}^{\bar{\pi}}(h \cdot a \cdot o') \right. \\
&\quad \quad \quad \left. - \mathbb{E}_{\bar{s} \sim \bar{\mathcal{T}}^*(h)} \mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot | \bar{s}, a)} \mathbb{E}_{o' \sim \bar{\mathcal{O}}(\cdot | \bar{s}')} \bar{V}^{\bar{\pi}}(h \cdot a \cdot o') \right| \\
&\quad \quad \quad \text{(Jensen's inequality)} \\
&\leq \gamma \cdot \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, a)} \mathbb{E}_{o' \sim \mathcal{O}(\cdot | s', a)} \left[V^{\bar{\pi}}(h \cdot a \cdot o') - \mathbb{E}_{\hat{o}' \sim \bar{\mathcal{O}}(\cdot | \phi(s', o'))} \bar{V}^{\bar{\pi}}(h \cdot a \cdot \hat{o}') \right] \right| \\
&\quad + \gamma K_{\bar{V}} L_{\mathbf{P}} \\
&\quad + \gamma \cdot \mathbb{E}_{h, o \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{\bar{s} \sim \Phi^*(h)} K_{\bar{V}} \mathcal{W}_{\bar{d}}(\bar{\mathbf{P}}(\cdot | \phi(s, o), a), \bar{\mathbf{P}}(\cdot | \bar{s}, a)) \\
&\quad + \gamma \cdot \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{\bar{s} \sim \Phi^*(h)} \mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot | \bar{s}, a)} \mathbb{E}_{o' \sim \bar{\mathcal{O}}(\cdot | \bar{s}')} \bar{V}^{\bar{\pi}}(h \cdot a \cdot o') \right. \\
&\quad \quad \quad \left. - \mathbb{E}_{\bar{s} \sim \bar{\mathcal{T}}^*(h)} \mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot | \bar{s}, a)} \mathbb{E}_{o' \sim \bar{\mathcal{O}}(\cdot | \bar{s}')} \bar{V}^{\bar{\pi}}(h \cdot a \cdot o') \right| \\
&\quad \quad \quad \text{(by Lemma 26)} \\
&= \gamma \cdot \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, a)} \mathbb{E}_{o' \sim \mathcal{O}(\cdot | s', a)} \left[V^{\bar{\pi}}(h \cdot a \cdot o') - \mathbb{E}_{\hat{o}' \sim \bar{\mathcal{O}}(\cdot | \phi(s', o'))} \bar{V}^{\bar{\pi}}(h \cdot a \cdot \hat{o}') \right] \right| \\
&\quad + \gamma K_{\bar{V}} \cdot (L_{\mathbf{P}} + L_{\bar{\mathbf{P}}}^{\Phi}) \\
&\quad + \gamma \cdot \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{\bar{s} \sim \Phi^*(h)} \left[\mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot | \bar{s}, a)} \mathbb{E}_{o' \sim \bar{\mathcal{O}}(\cdot | \bar{s}')} \bar{V}^{\bar{\pi}}(h \cdot a \cdot o') \right] \right. \\
&\quad \quad \quad \left. - \mathbb{E}_{\bar{s} \sim \bar{\mathcal{T}}^*(h)} \left[\mathbb{E}_{\bar{s}' \sim \bar{\mathbf{P}}(\cdot | \bar{s}, a)} \mathbb{E}_{o' \sim \bar{\mathcal{O}}(\cdot | \bar{s}')} \bar{V}^{\bar{\pi}}(h \cdot a \cdot o') \right] \right| \\
&\quad \quad \quad \text{(by definition of } L_{\bar{\mathbf{P}}}^{\Phi}, \text{ Equation 4.6)}
\end{aligned}$$

$$\begin{aligned}
 &\leq \gamma \cdot \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, a)} \mathbb{E}_{o' \sim \mathcal{O}(\cdot | s', a)} \left[V^{\bar{\pi}}(h \cdot a \cdot o') - \mathbb{E}_{\hat{o}' \sim \bar{\mathcal{O}}(\cdot | \phi(s', o'))} \bar{V}^{\bar{\pi}}(h \cdot a \cdot \hat{o}') \right] \right| \\
 &\quad + \gamma K_{\bar{V}} \cdot (L_{\mathbf{P}} + L_{\bar{\mathbf{P}}}^{\Phi}) \\
 &\quad + \gamma \cdot \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} K_{\bar{V}} \mathcal{W}_{\bar{d}}(\bar{\mathcal{T}}^*(h), \Phi^*(h)) \quad (\text{by Lemma 26}) \\
 &\leq \gamma \cdot \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, a)} \mathbb{E}_{o' \sim \mathcal{O}(\cdot | s', a)} \left[V^{\bar{\pi}}(h \cdot a \cdot o') - \mathbb{E}_{\hat{o}' \sim \bar{\mathcal{O}}(\cdot | \phi(s', o'))} \bar{V}^{\bar{\pi}}(h \cdot a \cdot \hat{o}') \right] \right| \\
 &\quad + \gamma K_{\bar{V}} \cdot (L_{\mathbf{P}} + L_{\bar{\mathbf{P}}}^{\Phi} + L_{\bar{\mathcal{T}}}) \quad (\text{by definition of } L_{\bar{\mathcal{T}}}, \text{Equation 4.6}) \\
 &= \gamma \cdot \mathbb{E}_{h, o \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{s', o' \sim \mathbf{P}_{\Omega}(\cdot | s, o, a)} \left[(V^{\bar{\pi}}(h \cdot a \cdot o') - \bar{V}^{\bar{\pi}}(h \cdot a \cdot o')) \right. \right. \\
 &\quad \left. \left. + \left(\bar{V}^{\bar{\pi}}(h \cdot a \cdot o') - \mathbb{E}_{\hat{o}' \sim \bar{\mathcal{O}}(\cdot | \phi(s', o'))} \bar{V}^{\bar{\pi}}(h \cdot a \cdot \hat{o}') \right) \right] \right| \\
 &\quad + \gamma K_{\bar{V}} \cdot (L_{\mathbf{P}} + L_{\bar{\mathbf{P}}}^{\Phi} + L_{\bar{\mathcal{T}}}) \\
 &\leq \gamma \cdot \mathbb{E}_{h, o \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{s', o' \sim \mathbf{P}_{\Omega}(\cdot | s, o, a)} [V^{\bar{\pi}}(h \cdot a \cdot o') - \bar{V}^{\bar{\pi}}(h \cdot a \cdot o')] \right| \\
 &\quad + \gamma \cdot \mathbb{E}_{h, o \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{s', o' \sim \mathbf{P}_{\Omega}(\cdot | s, o, a)} \left[\bar{V}^{\bar{\pi}}(h \cdot a \cdot o') - \mathbb{E}_{\hat{o}' \sim \bar{\mathcal{O}}(\cdot | \phi(s', o'))} \bar{V}^{\bar{\pi}}(h \cdot a \cdot \hat{o}') \right] \right| \\
 &\quad + \gamma K_{\bar{V}} \cdot (L_{\mathbf{P}} + L_{\bar{\mathbf{P}}}^{\Phi} + L_{\bar{\mathcal{T}}}) \quad (\text{triangle inequality}) \\
 &\leq \gamma \cdot \mathbb{E}_{h, o \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{s', o' \sim \mathbf{P}_{\Omega}(\cdot | s, o, a)} [V^{\bar{\pi}}(h \cdot a \cdot o') - \bar{V}^{\bar{\pi}}(h \cdot a \cdot o')] \right| \\
 &\quad + \gamma \cdot \mathbb{E}_{h, o \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, a)} \left| \mathbb{E}_{o' \sim \mathcal{O}(\cdot | s', a)} \left[\bar{V}^{\bar{\pi}}(h \cdot a \cdot o') - \mathbb{E}_{\hat{o}' \sim \bar{\mathcal{O}}(\cdot | \phi(s', o'))} \bar{V}^{\bar{\pi}}(h \cdot a \cdot \hat{o}') \right] \right| \\
 &\quad + \gamma K_{\bar{V}} \cdot (L_{\mathbf{P}} + L_{\bar{\mathbf{P}}}^{\Phi} + L_{\bar{\mathcal{T}}}) \quad (\text{by definition of } \mathbf{P}_{\Omega} \text{ and the Jensen's inequality}) \\
 &\leq \gamma \cdot \mathbb{E}_{h, o \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{s', o' \sim \mathbf{P}_{\Omega}(\cdot | s, o, a)} [V^{\bar{\pi}}(h \cdot a \cdot o') - \bar{V}^{\bar{\pi}}(h \cdot a \cdot o')] \right| \\
 &\quad + \gamma \cdot \mathbb{E}_{h, o \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{s' \sim \mathbf{P}(\cdot | s, a)} K_{\bar{V}} d_{TV} \left(\mathcal{O}(\cdot | s', a), \mathbb{E}_{o' \sim s', a} \bar{\mathcal{O}}(\cdot | \phi(s', o')) \right) \\
 &\quad + \gamma K_{\bar{V}} \cdot (L_{\mathbf{P}} + L_{\bar{\mathbf{P}}}^{\Phi} + L_{\bar{\mathcal{T}}}) \quad (\text{cf. Lemmae 25 and 26})
 \end{aligned}$$

$$\begin{aligned}
&= \gamma \cdot \mathbb{E}_{h, o \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \left| \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{s', o' \sim \mathbf{P}_{\Omega}(\cdot | s, o, a)} \left[V^{\bar{\pi}}(h \cdot a \cdot o') - \bar{V}^{\bar{\pi}}(h \cdot a \cdot o') \right] \right| \\
&\quad + \gamma K_{\bar{V}} \cdot (L_P + L_P^{\Phi} + L_{\bar{T}} + L_{\mathcal{O}}) \quad (\text{by definition of } L_{\mathcal{O}}, \text{ Equation 4.5}) \\
&\leq \gamma \cdot \mathbb{E}_{h, o \sim \mathcal{H}_{\bar{\pi}}} \mathbb{E}_{a \sim \bar{\pi}(\cdot | \Phi^*(h))} \mathbb{E}_{s \sim \mathcal{T}^*(h)} \mathbb{E}_{s', o' \sim \mathbf{P}_{\Omega}(\cdot | s, o, a)} \left| V^{\bar{\pi}}(h \cdot a \cdot o') - \bar{V}^{\bar{\pi}}(h \cdot a \cdot o') \right| \\
&\quad + \gamma K_{\bar{V}} \cdot (L_P + L_P^{\Phi} + L_{\bar{T}} + L_{\mathcal{O}}) \quad (\text{Jensen's inequality}) \\
&= \gamma \cdot \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} \left| V^{\bar{\pi}}(h) - \bar{V}^{\bar{\pi}}(h) \right| + \gamma K_{\bar{V}} \cdot (L_P + L_P^{\Phi} + L_{\bar{T}} + L_{\mathcal{O}}) \\
&\quad (\text{stationary property of } \mathcal{H}_{\bar{\pi}})
\end{aligned}$$

Putting all together. To recap, by Part 1 and 2, we have:

$$\begin{aligned}
\mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} |V^{\bar{\pi}}(h) - \bar{V}^{\bar{\pi}}(h)| &\leq L_R + L_{\bar{R}}^{\Phi} + R_{\text{MAX}} L_{\bar{T}} + \gamma \cdot \mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} |V^{\bar{\pi}}(h) - \bar{V}^{\bar{\pi}}(h)| \\
&\quad + \gamma K_{\bar{V}} \cdot (L_P + L_P^{\Phi} + L_{\bar{T}} + L_{\mathcal{O}}) \\
\mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} |V^{\bar{\pi}}(h) - \bar{V}^{\bar{\pi}}(h)| \cdot (1 - \gamma) &\leq L_R + L_{\bar{R}}^{\Phi} + R_{\text{MAX}} L_{\bar{T}} + \gamma K_{\bar{V}} \cdot (L_P + L_P^{\Phi} + L_{\bar{T}} + L_{\mathcal{O}}) \\
\mathbb{E}_{h \sim \mathcal{H}_{\bar{\pi}}} |V^{\bar{\pi}}(h) - \bar{V}^{\bar{\pi}}(h)| &\leq \frac{L_R + L_{\bar{R}}^{\Phi} + R_{\text{MAX}} L_{\bar{T}} + \gamma K_{\bar{V}} \cdot (L_P + L_P^{\Phi} + L_{\bar{T}} + L_{\mathcal{O}})}{1 - \gamma}
\end{aligned}$$

which finally concludes the proof. $\square$

C.2 Hyperparameters

Table 5 provides the range of hyperparameters used in the search, along with the selected values for each environment. The hyperparameter search was performed using OPTUNA [Akiba et al., 2019]. Pre-training of the WAE-MDP involved collecting 10240 transitions with a random policy and performing 200 training steps. These pre-training transitions are taken into account in the reported results.

Additionally, Table 6 (for R-A2C) and Table 7 (for DVRL) present the specific hyperparameters used for each algorithm. A grid search was conducted over all possible combinations for both baselines. The hidden size of all neural networks was set to 128 neurons and two hidden layers (except for the sub-belief encoder, and the policy which uses one) without further tuning. The experiments were carried out using 16 parallel environments. The original implementation of DVRL and their version of R-A2C were used in this study.

We ran the experiments on a cluster composed of Intel Xeon Gold 6148 CPU.

C. THE WASSERSTEIN BELIEVER: SUPPLEMENTARY MATERIAL

In the following tables the environments are represented as follows: STATELESSCARTPOLE ♥, NOISYSTATELESSCARTPOLE ♣, REPEATPREVIOUS ♦, SPACEINVADERS ♠, NOISYSPIGSPACEINVADERS ★.

The t-SNE [van der Maaten and Hinton, 2008] presented in the main text, in Figure 4.9, was obtained using cuML [Raschka et al., 2020]. Table 8 reports the hyperparameters that were used. We used 50000 sub-belief-value pairs, sampled from a dataset of 1 million timesteps collected at the end of the training.

Table 5: Range of hyperparameter search and selection per environment.

	Range	♥	♣	♦	♠	★
WAE Updates per Belief Update	1-2	1	1	2	2	2
Activation function	leaky relu, elu	leaky relu	leaky relu	elu	elu	relu
Activation function lipshitz	leaky relu, tanh	tanh	tanh	tanh	tanh	leaky relu
Activation CNN	leaky relu, elu				elu	elu
Policy config						
Learning rate	1.e-4, 3.e-4, 5.e-4	3.e-4	1.e-4	1.e-4	5.e-4	5.e-4
Clip norm	1, 10	10	10	10	10	10
Belief config						
Same opt as policy	Yes, No	Yes	Yes	Yes	No	No
Learning rate	1.e-4, 3.e-4, 5.e-4				1.e-4	3.e-4
Loss factor	1.e-1, 1.e-2, 1.e-3, 1.e-4, 1.e-5	1.e-1	1.e-4	1.e-5		
Filter variance min	1.e-2, 1.e-3	1.e-2	1.e-2		1.e-2	1.e-2
Normalize log obs filter	True, False	False	False	True	True	True
Sub belief prior temperature	.5, 0.66, .75, .9, .99	0.66	0.9	0.66	0.99	0.99
Reward loss scale factor	0.1, 1., 10., 20., 50., 100.	0.1	20	100	100	100
Transition loss scale factor	0.1, 1., 10., 20., 50., 100.	50	100	50	50	100
Buffer size	4096, 8192, 16384	4096	4096	16384	4096	4096
N Critic Update	5, 10	10	10	5	5	5
N State samples	16, 32, 64	32	64	32	32	32
N next state samples	16, 32, 64	32	64	64	16	32
WAE config						
Latent state size	♥, ♣ : 5 → 10, Others: 18 → 25	5	6	20	18	19
Minimizer learning rate	1.e-3, 3.e-4, 1.e-4, 5.e-5, 1.e-5	5.e-5	5.e-5	3.e-4	1.e-5	1.e-5
Maximizer learning rate	1.e-3, 3.e-4, 1.e-4, 5.e-5, 1.e-5	5.e-5	5.e-5	1.e-3	3.e-4	1.e-4
State encoder temperature	0.33, .5, 0.66, .75, .9, .99	0.33	0.75	0.66	0.5	0.5
State prior temperature	0.33, .5, 0.66, .75, .9, .99	0.75	0.75	0.5	0.75	0.75
Local transition loss scaling	10., 25., 50., 75., 80.	50	80	80	10	10
Steady state scaling	10., 25., 50., 75., 80.	80	80	100	25	25
N critic update	5, 10	10	10	5	5	10
Batch size	128, 256	128	128	256	128	128
Clip grad	1, 10, 100.	100	100	100	10	10
State vs Obs reconstruction weight	1, 2	2	2	2	1	1
Obs reg. min learning rate	1.e-3, 3.e-4, 1.e-4, 5.e-5, 1.e-5	5.e-5	1.e-4	1.e-3	5.e-5	5.e-5
Obs reg. max learning rate	1.e-3, 3.e-4, 1.e-4, 5.e-5, 1.e-5	5.e-5	3.e-4	5.e-5	1.e-5	3.e-4
Obs reg. gradient penalty	50, 100, 500, 1000	50	50	1000	100	1000

Table 6: R-A2C hyperparameters

	Range	♥	♣	♦	♠	★
Optimizer	Adam, RMSProp	RMSProp	RMSProp	Adam	RMSProp	RMSProp
Clip grad norm	0.5, 1., 10.	1.0	10	10.	10	10
Learning rate	3.e-5, 1.e-4, 3.e-4, 5.e-4, 1.e-3	5.e-4	1.e-4	5.e-4	1.e-3	1.e-3

Table 7: DVRL hyperparameters

	Range	♥	♣	♦	♠	★
Optimizer	Adam, RMSProp	Adam	Adam	RMSProp	RMSProp	RMSProp
Clip grad norm	0.5, 1., 10.	10.0	1.0	1.0	10	0.5
Learning rate	3.e-5, 1.e-4, 3.e-4, 5.e-4, 1.e-3	5.e-4	1.e-3	3.e-4	1.e-3	1.e-3
Encoding loss factor	1., 1., 5., 05	0.5	1.0	1.0	1.0	0.05
Number of particles	5, 10, 15	10	3	10	5	5

Table 8: t-SNE hyperparameters

Method	FFT
Perplexity	10
N Neighbors	50
Metric	Manhattan
Learning rate	2400
Early exaggeration	50
Late exaggeration	3
Exaggeration iter	300

D Synthesis from DRL components

D.1 Details on the Experiments

In this section, we give additional information on the setting used during the experiments presented in Chapter 5.

Grid world. Figure 1 shows a 3×3 grid environment where each room is composed of 20×20 cells.

Hyperparameter search. To train our WAE-DQN agent, we ran 4 environments in parallel per training instance and used a replay buffer of capacity $7.5 \cdot 10^5$. We performed a grid search to find the best parameters for our WAE-DQN algorithm. Table 9 presents the range of hyperparameters used. In particular, we found that prioritized experience

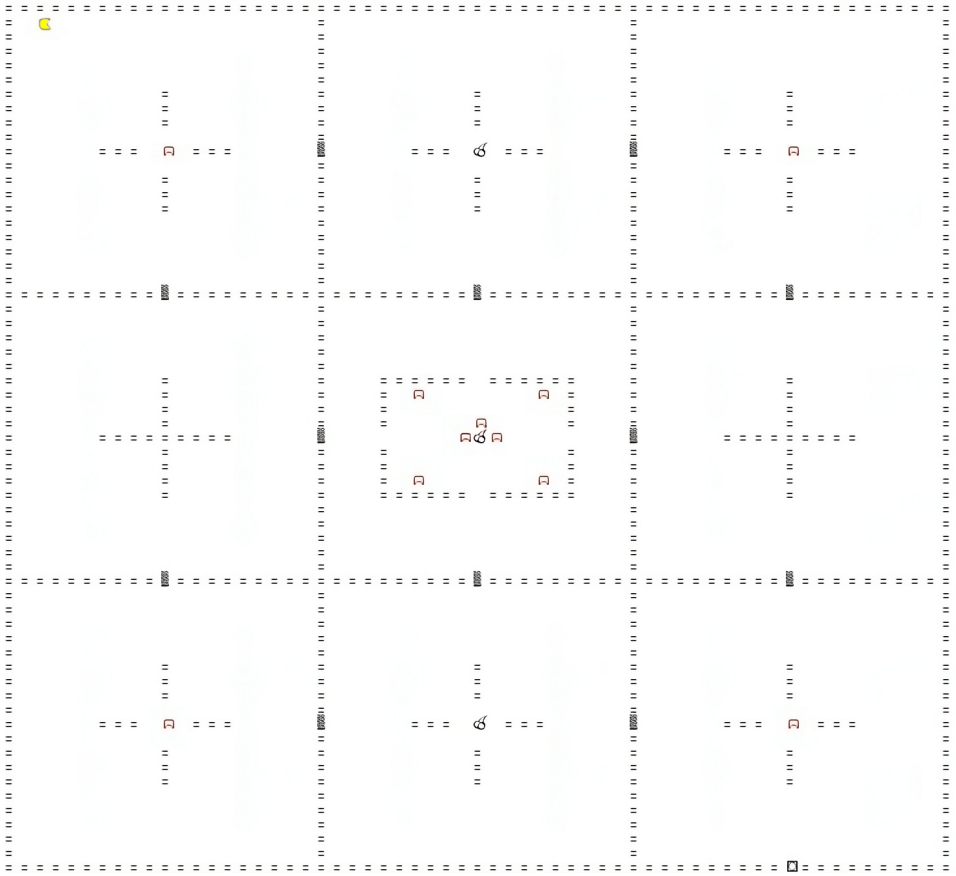


Figure 1: Environment for $N = 9$ rooms of 20×20 cells. The agent is depicted in yellow (top left), adversaries in red, power-ups as cherries, and the goal at the bottom right.

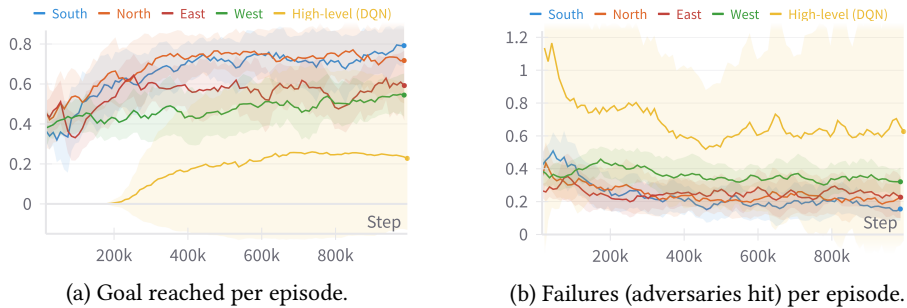


Figure 2: A more detailed version of Figure 5.8, where the WAE-DQN performance is specified per direction.

replay Schaul et al. [2016] and a categorical Q -network Bellemare et al. [2023] did not improve the results in our grid-world environment significantly. We used a batch size of 128 for the WAE-MDP.

For synthesizing the high-level controller, we used the hyperparameters that worked the best for each specific direction. We used the same parameters for the DQN training instances shown in Figure 5.8. Additionally, with Figure 2, we provide a more detailed version of the figure, showing the results of each latent policy for each specific direction.

For the MADE to model the latent entrance function, we used a dataset of size 25600, and the training was split into 100 epochs (i.e., the model performed 100 passes through the entire dataset) with a learning rate of 10^{-3} . We used a batch size of 32 or 64, and two hidden layers, either with 64 or 128 neurons.

Table 9: Hyperparameter range used for (WAE-)DQN. Parameters in green worked best on average. For details about the WAE-MDP parameters, see Appendix B.

Parameter	Value
Common to DQN and WAE-MDP	
Activation	{ReLU, leaky ReLU, ELU, tanh, sigmoid}
# Hidden layers per network	{1, 2, 3}
# Neurons per layer	{128, 256, 512}
CNN filters (3 layers)	{3 → 5 → 7, 3 → 3 → 3}
CNN kernels (3 layers)	{32 → 64 → 16, 64 → 32 → 16}
DQN	
Use Boltzmann exploration	{Yes, No}
Boltzmann temperature	{0.25, 0.10, 0.75, 1, 10, 100}
Use ϵ -greedy exploration (decay to $\epsilon = 0.1$)	{Yes, No}
Target update period	{1, 250, 500, 1000}
Target update scale (α in Algorithm 8)	{ 10^{-4} , $5 \cdot 10^{-4}$, 10^{-3} , $5 \cdot 10^{-3}$ }
Reward scaling	{1, 10, 25, 100}
Learning rate	{ $6.25 \cdot 10^{-5}$, 10^{-4} , $2.5 \cdot 10^{-4}$, 10^{-3} }
Batch size	{32, 64, 128}
Use double Q -networks van Hasselt et al. [2016]	{Yes, No}
WAE-MDP	
Latent state size (power of 2)	{12, 13, 14, 15}
State embedding function temperature	{ $1/3$, $1/2$, $2/3$, $3/4$, 0.99}
Transition function temperature	{ $1/3$, $1/2$, $2/3$, $3/4$, 0.99}
Steady-state regularizer scale factor	{10, 25, 50, 75}
Transition regularizer scale factor	{10, 25, 50, 75}
Minimizer learning rate	{ 10^{-4} , $5 \cdot 10^{-4}$, 10^{-3} }
Maximizer learning rate	{ 10^{-4} , $5 \cdot 10^{-4}$, 10^{-3} }
State embedding function learning rate	{ 10^{-4} , $5 \cdot 10^{-4}$, 10^{-3} }
# critic updates	{5, 10, 15}
State reconstruction function	{ L_2 , binary cross entropy (for M)}

Index

A

- A2C, 74
- abstraction quality, 141
 - PAC bounds, 148, 158
- acceptance condition, 86
- action lifting, 137
- activation function, 55
 - ReLU, 55
 - sigmoid, 55
- actor, 70
 - critic, 73
- Adam, 56
- adaptive learning rate, 55
- advantage, 73
- almost sure event, 28
- aperiodicity, 43, 45
- autoregressive density, 194
- average
 - episode length (AEL), 72, 149
 - reward, 56

B

- base, 37
- batch, 55
- Bayes rule, 31
- behavioral
 - model, 161
 - policy, 54
 - ϵ -greedy, 65

- Boltzmann, 65
- belief, 213
 - MDP, 214
 - update function, 214
- Bellman
 - equation, 47
 - error, 50
 - MSBE, 66
 - operator, 49, 59
- Bernoulli, 171
 - multivariate, 194
 - relaxed, 172
- bias, 55
- bisimulation
 - between MDPs, 119
 - bounds, 140
 - distance, 130
 - fixed point interpretation, 129
 - kernel, 130
 - pseudometric, 130
 - quotient, 123
 - relation, 119
- Boltzmann
 - policies, 65
 - temperature, 65
- bootstrapping, 73
- Borel measure, 29
- BSCC, 41

C

cells, 55
compact plan, 258
 latent, 263
consistent histories, 228
continual RL, 56
contraction mapping, 50
coupling, 125
critic, 73
cylinder set, 37

D

decoder, 163, 165, 173
deep
 MDPs, 138
 RL (DRL), 55
density function, 30
Dirac, 28, 167, 213
direction, 245
discounted properties, 101
discrete distribution, 29
distance, 124
distillation, 161
distortion, 164
DKL, 124
DQN, 66
 rainbow, 70
 target
 network, 68
 value, 66
DRA, 87
 acceptance condition, 86
drift, 75

E

earth mover's distance, 127
edge, 246
ELBO, 164
 local, 168
enabled actions (in an EC), 91
encoder, 161, 164, 193

end component (EC), 91
entrance
 function, 245
 latent, 264
 loss, 264
entropy, 162
 regularization, 173
episode, 57
episodic
 process, 57
 return, 57
 value difference bound, 145
episodic process, 56
epsilon (ϵ)-greedy, 65
epsilon (ϵ)-perturbation trick, 62
equivalence class, 118
ergodicity, 43, 45, 61
event, 28
 almost sure, 28
 measurable, 28
exit function, 245
expectation, 30
 conditional, 31
expected return, 46
experience replay, 68
 prioritized, 70, 175
exploration vs. exploitation, 54

F

feed-forward network, 55

G

generalized policy iteration, 70
gradient step, 55
graph, 246
 neighbors, 246
Gumbel
 distribution, 170
 samples, 170
 softmax, 170

H

Harris chain, 44
 recurrent, 44
 hidden layers, 55
 hierarchical controller, 248
 hierarchical model, 245, 246
 high-level map, 245
 history (in a POMDP), 213
 consistent, 228
 unfolding, 219
 Hoeffding's inequality, 155
 Hutchinson, 127

I

importance sampling, 76, 175
 weight, 175
 infinitely often (visited states), 42
 integral probability metric (IPM), 127
 irreducibility, 43

J

Jensen's inequality, 144

K

Kantorovich, 126
 Kullback-Leibler (KL) divergence, 124

L

latent
 belief, 222
 belief encoder, 218
 compact plan, 263
 entrance function, 263, 264
 MDP, 137
 policy, 137
 POMDP, 217
 space model, 137
 learning rate, 55
 lifted latent policy, 222
 linear program, 48, 85
 Lipschitz

function, 126

liveness, 108
 local losses, 140
 location-scale distribution, 172
 logistic
 distribution, 172
 function, 176
 loss function, 55
 LTL, 79
 model checking, 94
 PAC learnability, 97
 semantics, 80
 syntax, 80
 synthesis, 93

M

MADE, 195
 MAF, 194
 Markov chain (MC), 32
 (a)periodic, 43, 45
 BSCC, 41
 Harris recurrent, 44
 irreducible, 43
 limiting behaviors, 40
 occupancy measure, 40
 stationary distribution, 40
 transient distribution, 40
 with restarts, 45
 Markov's inequality, 147
 maximal end component (MEC), 92
 MDP, 32, 35
 continuous, 32
 episodic, 56
 partially observable, 207, 213
 plan, 253
 product with DRA, 89
 terminal, 58
 unknown, 54
 Mealy machine, 34
 mean

- payoff, 56
- recurrence time, 149
- memory of a policy, 34
- meta-information, 69
- metric space, 124
- mini-batch, 55
- mirror descent, 70
- model checking, 78, 82
 - LTL, 88, 94
 - VLC, 111
- Monge, 127
- monotone mapping, 50
- Monte Carlo estimation, 55
- MSBE, 66

N

- neighbors (graph), 246
- neural network, 55
 - feed-forward, 55
- neurons, 55
- normal distribution, 170

O

- observation loss, 225
- occupancy measure, 40
- omega (ω)-regular properties, 86
- on/off-policy algorithm, 54
- optimal transport (OT), 125
- optimizer, 56
 - Adam, 56
 - RMSprop, 56
- outcome, 28

P

- PAC, 65
 - learnability, 99
- parity
 - automaton, 112
 - condition, 113
- path (graph), 246
- PER, 70, 175

- bucket-based, 175
- loss-based, 175
- periodicity, 43
- persistence, 108
- planner, 248
- policy, 34
 - behavioral, 54
 - encoded as a Mealy machine, 34
 - gradient, 71
 - latent, 137
 - Markovian, 34
 - memory, 34
 - proper, 255
 - stationary, 34
- policy gradient theorem, 71
- policy iteration (PI), 52, 86
 - generalized, 70
- POMDP, 207, 213
 - refined, 216
- Portmanteau theorem, 167
- posterior collapse, 173
- posterior distribution, 163
- PPO, 75
- prefix (of a trajectory), 37
- priority exponent, 175
- probability
 - measure, 28
 - space, 28
- product of an MDP with a DRA, 89
- proper policy, 255
- pseudo-polynomial, 51
- pseudometric, 128

Q

Q-

- learning, 64
- network, 66
- table, 64
- value, 46

R

Rabin
 automaton, 87
 chain, 112
 random variable, 29
 density, 30
 expected value, 30
 variance, 30
 rate, 164
 raw transition metric, 182
 reach-avoid objectives, 245
 reachability
 discounted, 105
 high-level, 247
 in LTL, 82
 in VLC, 105
 recurrent Harris chain, 44
 relaxed Bernoulli distribution, 172
 ReLU, 55
 reparameterization trick, 170
 representation quality, 152
 reset, 56
 return, 46
 RL, 54
 continual, 56
 episodic, 56
 RMSprop, 56
 RNN, 209
 room, 245
 Rubinstein, 126

S

safety, 106
 low-level, 247
 sample space, 28
 SGD, 55
 sigma (σ)-algebra, 28
 sigmoid, 55
 soft actor-critic (SAC), 73
 softmax, 65

sparse rewards, 53
 state embedding, 137
 stationary distribution, 40
 over histories, 219
 steady-state regularizer, 187
 stochastic shortest path problem, 84
 straight-through gradients, 193
 sub-belief, 232
 encoder, 233
 sufficient statistic, 214
 supremum norm, 50
 syntactically contractive operator, 102
 synthesis, 79
 from deep RL policies, 241
 LTL, 93
 reachability, 85

T

target network, 68
 update rate, 68
 temperature
 Boltzmann, 65
 relaxed distributions, 170
 terminal
 MDP, 58
 state, 58
 total variation, 127
 trace, 33
 distribution, 162
 trajectory, 33
 prefix, 37
 transient distribution, 40
 transition regularizer, 187
 transport plan, 125
 TRPO, 75

U

undiscounted return, 58, 72, 84
 unknown environment, 54

INDEX

V

VAE-MDP, 165
value function, 46
 episodic, 57
 in VLC, 103
value iteration (VI), 49, 86
variable valuation, 103
variance, 30
variational
 architecture, 164
 inference, 163
vertex, 246
VLC, 101
 linear properties, 105, 109

liveness and persistence, 108
model checking, 111
reachability, 105
safety, 106
semantics, 103
synthax, 102

W

WAE, 183
 -MDP, 185
 objective, 187
Wasserstein, 126
WBU, 211
weights, 55