BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INTELLIGENT SYSTEMS
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

IMPROVING SYNTHESIS OF FINITE STATE CONTROLLERS FOR POMDPS USING BELIEF SPACE APPROXIMATION

VYLEPŠENÍ SYNTÉZY KONEČNĚ STAVOVÝCH KONTROLÉRŮ PRO POMDP

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR **Bc. FILIP MACÁK**
AUTOR PRÁCE

SUPERVISOR **doc. RNDr. MILAN ČEŠKA, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2023

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Master's Thesis Assignment

146891

Institut:	Department of Intelligent Systems (UITS)
Student:	**Macák Filip, Bc.**
Programme:	Information Technology and Artificial Intelligence
Specialization:	Mathematical Methods
Title:	**Improving Synthesis of Finite State Controllers for POMDPs Using Belief Space Approximation**
Category:	Formal Verification
Academic year:	2022/23

Assignment:

1. Study the state-of-the-art controller synthesis methods for Partially Observable MDPs (POMDPs) with the focus on inductive synthesis and belief space approximation supporting complex indefinite-horizon specifications.
2. Design an integrated controller synthesis method combining benefits of inductive synthesis and belief space approximation.
3. Implement the integrated method within the tool PAYNT.
4. Using suitable benchmarks, perform a detailed experimental evaluation of the integrated method including a comparison with the state-of-the-art synthesis methods.

Literature:

- Kochenderfer, M.J., Wheeler, T.A., and Wray K.H, Algorithms for Decision Making, MIT Press 2021.
- Andriushchenko, R., Češka, M., Junges, S., and Katoen, J.P. Inductive synthesis of finite-state controllers for POMDPs. In UAI'22. Proceedings of Machine Learning Research.
- Bork, A., Katoen, J.P, and Quatmann, T. Under-approximating expected total rewards in POMDPs. In TACAS'22. Springer.
- Andriushchenko, R., Češka, M., Junges, S., Katoen, J.P. and Stupinský, Š. PAYNT: A Tool for Inductive Synthesis of Probabilistic Programs. In *CAV 2021*. Springer.

Requirements for the semestral defence:
Items 1, 2 and partially item 3.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Češka Milan, doc. RNDr., Ph.D.**
Consultant:	Andriushchenko Roman, Ing.
Head of Department:	Hanáček Petr, doc. Dr. Ing.
Beginning of work:	1.11.2022
Submission deadline:	24.5.2023
Approval date:	3.11.2022

Abstract

This work focuses on combining two state-of-the-art controller synthesis methods for partially observable Markov decision processes (POMDPs), a prominent model in sequential decision making under uncertainty. A central issue is to find a POMDP controller that achieves a total expected reward objective. As finding optimal controllers is undecidable, we concentrate on synthesising good finite-state controllers (FSCs). We do so by tightly integrating two modern, orthogonal methods for POMDP controller synthesis: a belief-based and an inductive approach. The former method obtains an FSC from a finite fragment of the so-called belief MDP, an MDP that keeps track of the probabilities of equally observable POMDP states. The latter is an inductive search technique over a set of FSCs with a fixed memory size. The key result of this work is a symbiotic anytime algorithm that tightly integrates both approaches such that each profits from the controllers constructed by the other. Experimental results indicate a substantial improvement in the value of the controllers while significantly reducing the synthesis time and memory footprint.

Abstrakt

Táto práca sa zameriava na kombináciu dvoch moderných metód syntézy plánovačov pre Markovské procesy s čiastočným pozorovaním (POMDPs), ktoré sú významným modelom pre sekvenčné rozhodovanie s neistotou. Hlavnou úlohou je nájsť plánovač POMDP, ktorý dosahuje čo najlepšiu hodnotu. Keďže hľadanie optimálneho plánovača je nerozhodnuteľné, zameriavame sa na syntézu dobrých konečne stavových kontrolérov (FSCs). V tejto práci integrujeme dve moderné, ortogonálne metódy pre syntézu kontrolérov POMDP, a to metódu založenú na prehľadávaní belief priestoru a induktívnu metódu. Prvá metóda získava FSC z konečného fragmentu takzvaného belief MDP, čo je MDP, ktorý udržiava prehľad o pravdepodobnostiach rovnako pozorovateľných stavov POMDP. Druhá je induktívna vyhľadávacia technika pre množinu FSC s fixnou veľkosťou pamäti. Kľúčovým výsledkom tejto práce je symbiotický algoritmus, ktorý integruje obidva tieto prístupy tak, aby sa každý dokázal zlepšiť z kontrolérov vytvorených tým druhým. Experimentálne výsledky naznačujú významné zlepšenie hodnoty kontrolérov pri značnom znižovaní času syntézy a využitej pamäte.

Keywords

Markov models, probabilistic models, automated synthesis, model checking, formal methods, partial observability

Klíčová slova

Markovské modely, pravdepodobnostné modely, automatizovaná syntéza, model checking, formálne metódy, čiastočná pozorovateľnosť

Reference

MACÁK, Filip. *Improving Synthesis of Finite State Controllers for POMDPs Using Belief Space Approximation.* Brno, 2023. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor doc. RNDr. Milan Češka, Ph.D.

Rozšířený abstrakt

Náhodnosť a neistota sa často vyskytujú v umelej inteligencii, biologických systémoch, distribuovaných algoritmoch a mnohých ďalších oblastiach. Pravdepodobnostné modely sú silným nástrojom, ktorý nám pomáha uvažovať o náhodnosti a neistote v systémoch. Použitie týchto modelov nám môže pomôcť ďalej analyzovať vlastnosti daných systémov formálnym spôsobom. Modely zodpovedajúce systémom z reálneho života môžu byť veľmi komplexné. Ich stavový priestor môže byť veľký a môžu obsahovať veľa častí s neistými informáciami, ktoré môžu byť pre daný model kritické. Preto je dôležité zlepšovať metódy, ktoré sa používajú na analýzu takýchto modelov.

Markovské rozhodovacie procesy (MDP) [25] sú jedným z najpoužívanejších modelov na získavanie informácií o systémoch obsahujúcich pravdepodobnostné vetvenia s nedeterministickými akciami. Sú využívané v automatizovanom plánovaní, plánovaní úloh a formálnej verifikácii. Pravdepodobnostné modelovacie nástroje, ako napríklad PRISM [22] a Storm [12], sú schopné efektívne nájsť riešenia pre MDP. Čiastočne pozorovateľné Markovské rozhodovacie procesy (POMDP) [24] predstavujú zovšeobecnenie MDP. Zdedili všetky vlastnosti klasických MDP, ale ich stavy nie sú plne pozorovateľné. To môže byť interpretované tak, že vieme identifikovať len určité aspekty stavov, napríklad farbu stavu, ale nie samotný stav. To dáva POMDP potrebnú silu na modelovanie určitých problémov. Napríklad si predstavme robota, ktorého senzory majú len čiastočnú spoľahlivosť. Toto môžeme modelovať pomocou POMDP. Hlavným cieľom riešenia POMDP je získať plánovač, teda plán na riešenie nedeterminizmu modelu pre daný cieľ. Rozdielom oproti MDP pri tomto probléme je, že POMDP plánovače musia svoje rozhodnutia zakladať len na pozorovateľných aspektoch stavov, zatiaľ čo plánovače MDP môžu brať do úvahy celú históriu informácií o stavoch. To znamená, že v POMDP sú cesty s rovnakými sledmi pozorovaní neodlíšiteľné z pohľadu plánovača.

Problém overenia POMDP s neohraničeným horizontom (neobmedzená dosiahnuteľnosť alebo odmeny akcii) je vo všeobecnosti nerozhodnuteľný [23]. Jeden príklad takejto špecifikácie je nasledovný: *Je maximálna očakávaná celková odmena na dosiahnutie daného cieľového stavu v POMDP pod daným prahom?* Napriek nerozhodnuteľnosti existuje mnoho prístupov, ktoré sa dnes snažia účinne aproximovať riešenie pre POMDP. Belief-based metódy sú založené na preskúmaní belief priestoru a sú jedným z najmodernejších prístupov pre analýzu POMDP s neobmedzeným horizontom. "Belief" reprezentuje pravdepodobnostnú distribúciu nad stavmi POMDP, ktoré majú rovnaké pozorovanie. Pojem "belief" je dôležitý, pretože nám umožňuje opísať čiastočnú pozorovateľnosť aktuálneho stavu. Tieto metódy sa snažia odvinúť belief priestor daného POMDP a vytvoriť takzvaný belief MDP, ktorý je možné následne overiť. Dokážu tiež poskytnúť dobré výsledky pre väčšinu modelov, ale použitie plánovača, ktorý vytvoria, je komplikované, pretože tento plánovač obvykle vyžaduje veľa pamäte a spolieha sa na náhodné podplánovače. Preto vznikli metódy, ktoré dokážu produkovať malé a ľahko použiteľné plánovače. Jednou z týchto metód je induktívna syntéza konečne stavových kontrolérov (FSC) [4]. FSC sú malé konečné automaty, ktoré dokážu zakódovať celé stratégie pre POMDP [13]. Induktívna syntéza iteratívne preskúmava čoraz väčšie rodiny možných FSC a snaží sa nájsť ten najlepší. Jedným z problémov induktívnej syntézy je rast priestoru možných FSC, ktorý treba preskúmať. Táto práca sa teda sústreďuje na skúmanie a zlepšovanie dvoch hlavných metód: 1) induktívna syntéza FSC implementovaná v nástroji PAYNT [5], a 2) Belief-based metódy [6, 7] implementované v nástroji Storm [12, 17], ktoré používajú takzvané cut-off aproximácie nepreskúmaného belief priestoru. Obidve tieto metódy majú svoje výhody a nevýhody, preto z hľadiska používateľa nemusí byť jednoduché vybrať, ktorú z metód použiť bez hlbokých znalostí jednotlivých

metód a skúmaného modelu. Táto práca sa zameriava na vývoj frameworku, kde si tieto metódy symbioticky pomáhajú a sú schopné nájsť lepšie plánovače.

V tejto práci sa budeme zameriavať iba na špecifikácie s neobmedzeným horizontom. Ak odstránime túto požiadavku, existujú aj iné veľmi efektívne metódy. Významné príklady sú aproximovaná iterácia hodnôt navrhnutá v [15], simulácie Monte Carlo [27] a použitie strojového učenia a neurónových sietí [8]. Pravdepodobne najznámejší spomedzi týchto metód je algoritmus SARSOP [21], ktorý produkuje plánovače ako množinu α-vektorov. Plánovače vo forme α-vektorov vedú k zložitejšiemu následnému použitiu. Výsledné plánovače musia sledovať "belief" a vykonávať náročné výpočty na výber akcií. Ďalším významným algoritmom je HSVI [18]. Tieto metódy sú obvykle veľmi silné s využitím skutočnosti, že sa umožňuje znižovanie dôležitosti akcii. Niektoré prístupy sa uberali cestou hľadania samotných plánovačov na rozdiel od snahy získať plánovač s aproximovanej hodnotovej funkcie. Niektoré z najvýznamnejších sú náhodné plánovače pomocou gradientného zostupu [16] alebo pomocou konvexnej optimalizácie [1, 11, 19].

Reinforcement learning (RL) [14, 26] je tiež veľmi významným v probléme hľadania plánovačov pre POMDP (prehľad najnovších pokrokov v RL pre POMPD nájdete v [29]). Tieto prístupy a iné metódy strojového učenia zavádzajú istú polaritu medzi bezpečnosťou výsledku a škálovateľnosťou v porovnaní s formálnymi metódami, na ktoré sa zameriavame v tejto práci. Metódy založené na RL veľmi dobre škálujú a môžu byť tiež použité na riešenie neznámych POMDPs, ale obetujú záruky bezpečnosti, ktoré poskytujú formálne metódy. Cieľom tejto práce je zmierniť medzeru medzi bezpečnosťou a škálovateľnosťou, a to tak že sa snažíme zlepšiť efektívnosť formálnych metód na riešenie POMDP a posilniť ich pozíciu v tejto oblasti.

Hlavným prínosom tejto práce je symbióza belief-based metód a vyhľadávania plánovačov. Pre tento účel bolo potrebné vyriešiť rôzne technické prekážky, ako napríklad získanie plánovača z belief MDP s aproximujúcimi plánovačmi, a vývoj prechodu medzi fázami prieskumu belief priestoru a vyhľadávania plánovačov s minimálnymi nákladmi. Výhody navrhovaného symbiotického algoritmu sú mnohostranné, ako ukazujeme v našej empirickej evaluácii. Symbiotický algoritmus dokáže riešiť POMDP, ktoré nemožno riešiť žiadnym z jednotlivých prístupov samostatne. Produkuje plánovače, ktoré dosahujú lepšiu hodnotu (s relatívnym zlepšením až o 40%) ako aj plánovače, ktoré potrebujú menej pamäte (s redukciou až o dva rády). Okrem toho je táto integrácia schopná znížiť pamäťové nároky prieskumu belief priestoru až o faktor 4. Záver je taký, že naša integrácia ponúka veľmi silný syntetizačný algoritmus, ktorý produkuje lepšie a stručnejšie plánovače v porovnaní s najnovšími metódami. Zatiaľ čo naša empirická evaluácia je špecifická pre dané metódy, získané poznatky sa môžu aplikovať aj pri integrácii iných metód.

Improving Synthesis of Finite State Controllers for POMDPs Using Belief Space Approximation

Declaration

I hereby declare that this Master's thesis was prepared as an original work by the author under the supervision of doc. RNDr. Milan Češka, Ph.D. The supplementary information was provided by Ing. Roman Andriushchenko. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Filip Macák
May 15, 2023

Acknowledgements

I would like to thank my supervisor Milan Češka and the consultant for this work Roman Andriushchenko for their help and guidance during this research. I would also like to thank Alexander Bork, Sebastian Junges and Joost-Pieter Katoen for helping us with the creation of an article inspired by this work, that got accepted to CAV'23. I want to thank my parents for their support and encouragement throughout my whole life and especially during my time at the university. A big thanks goes to my girlfriend Janka for always being here for me and being an amazing part of my life. And last but not least, I would like to thank my friends Dizzax, Poizzi, Radluy and Toaster for all the good times that keep me going forward.

Contents

Chapter 1

Introduction

Randomness and uncertainty commonly appear in artificial intelligence, biological systems, distributed computing and many more fields. Probabilistic models are a powerful tool to help us reason about both randomness and uncertainty in systems. Using these models can help us further analyse the properties of given systems in a formal way. Models corresponding to many real-life systems can be very complex, their state space can be large and there can be a lot of parts where critical information is uncertain. That's why improving the methods that are used for the analysis of such models is important.

Markov decision processes (MDPs) [25] are one of the most used models to reason about systems containing probabilistic branching with controllable actions. They are used in automated planning, scheduling and formal verification. Probabilistic model checkers such as PRISM [22] and Storm [12] are able to efficiently find policies for MDPs. Partially observable Markov decision processes (POMDPs) [24] represent a generalization of MDPs. They inherit all features of classic MDPs, however, the states are not fully observable. This can be interpreted as only being able to identify certain aspects of states, but not the state itself. This gives the POMDPs the needed strength to model certain problems. For example, imagine a robot whose sensors have only partial reliability, we can model this using POMDPs. The main goal in POMDP solving is to obtain a scheduler, i.e. a plan for how to choose actions for a given objective. The difference for this problem between solving MDPs and POMDPs is that POMDP schedulers must base their decisions solely on the observable aspects of the states, meanwhile, MDP schedulers can take the entire history of full state information into consideration. This means that in POMDPs the paths with the same observation traces are indistinguishable for the scheduler.

The problem of verifying a POMDP with respect to indefinite-horizon specification (unbounded undiscounted reachability or rewards) is generally undecidable [23]. One example of such a specification is the following: *is the maximal expected total reward to reach a given goal state in a POMDP below a given threshold?* Despite the undecidability, there are many approaches today that try to effectively approximate the solution for POMDPs.

The belief-based methods using belief exploration are one of the state-of-the-art approaches for POMDPs [7]. Beliefs represent the probability distributions over states of the POMDP that have the same observation and thus allow us to describe the partial observability of the current state. Belief-based methods try to unfold the belief state space of a given POMDP and obtain policies from the explored belief space. These methods are able to provide high-quality schedulers, however, deploying such schedulers is problematic as they usually require a lot of memory, they are difficult to interpret since very similar beliefs can lead to different actions, and they rely on randomized sub-schedulers to approximate

unexplored belief space. Alternative approaches that can produce small and easy-to-use schedulers for complex POMDPs have recently emerged. One of the state-of-the-art approaches builds on the inductive synthesis of finite-state controllers (FSCs) [4]. FSCs are small finite-state automata that can encode whole strategies for POMDPs [13]. The inductive synthesis iteratively explores bigger and bigger families of possible FSCs and tries to find the best one. The key problem of inductive synthesis is the increasing size of the design space that needs to be explored. So the two main methods this work will try to examine and improve are: 1) the inductive synthesis of FSCs implemented in a tool called PAYNT [5], 2) the belief-based methods [6, 7] implemented in a tool called Storm [12, 17] that use cut-offs to approximate the unexplored belief space. Both of these methods have their pros and cons, so from the user's perspective, it is not convenient to choose between them without a deep understanding of the methods. This work sets out to develop a framework where these methods symbiotically alleviate each other's weaknesses and find better schedulers than both of them individually.

If we lift the need for undiscounted specification there are alternative methods that are very efficient. Prominent examples include approximative value iteration proposed in [15], Monte Carlo simulations [27] and usage of machine learning and neural networks [8]. Probably the most notable of the bunch is algorithm SARSOP [21] which produces policies as a set of α-vectors. The α-vector policies lead to more complex analysis downstream: the resulting policies must track the belief and do floating-point computations to select actions. Another notable algorithm is HSVI [18]. These methods are usually really strong in utilizing the fact that discounting is allowed. Some approaches go the route of policy search. Some of the most prominent are randomised controllers via gradient descent [16] or via convex optimization [1, 11, 19].

Reinforcement learning (RL) [14, 26] is also very prominent in the problem of finding controllers for POMDPs (for an overview of recent advances in RL for POMPDs see [29]). These approaches and other machine learning methods introduce a certain polarity between safety and scalability compared to the formal methods we focus on in this work. The RL-based methods scale very well and also might be used for solving unknown POMDPs, however, they sacrifice the safety guarantees offered by the formal methods. The focus of this work is on bridging the gap between safety and scalability as we aim to improve the efficiency of the formal methods for POMDP solving and strengthen their position in this research area.

Key contributions

This work showcases the issues and limitations of the state-of-the-art methods on tiny examples and uses this fact to reinforce the need for a symbiotic approach. These tiny examples provide a good insight into what the methods trying to tackle the indefinite-horizon specifications for POMDPs need to take into consideration to provide good results for a variety of problems.

The key contribution of this work is the symbiosis of belief exploration and policy search methods. Various technical obstacles had to be addressed e.g. obtaining a scheduler from the belief MDP along the approximating policies for its frontier and developing a back-and-forth approach that switches between the belief exploration and policy search phases with minimal overhead. The benefits of the symbiotic algorithm are manifold, as we show by in-depth empirical evaluation. The symbiotic algorithm is able to solve POMDPs that cannot be tackled with either of the two approaches alone. It produces schedulers that are superior

in value (with relative improvements of up to 40%) as well as schedulers which take a lot less memory to store (with a reduction of a factor of up to two orders of magnitude) with only a small cost on their values. Additionally, the integration is able to reduce the memory footprint of the belief exploration by a factor of 4. In conclusion, the integration offers a very powerful push-button, anytime synthesis algorithm producing superior and/or more succinct schedulers compared to the state-of-the-art methods. While our empirical evaluation is method-specific, the lessons carry over to integrating other methods.

Another important contribution is the unification of the theory behind representing the schedulers for POMDPs using FSCs. We define a notion of a general FSC and then we expand this definition and define various sub-classes of FSCs. This fact allows us to better reason about the schedulers produced by belief-based methods and also allows us to better compare the results, especially their size.

Publication

This thesis served as a basis for an article called „Search and Explore: Symbiotic Policy Synthesis in POMDPs“. This article was accepted to International Conference on Computer Aided Verification 2023 (CAV'23) in May 2023. CAV is a CORE A* conference with average acceptance rate around 25%. I would like to thank the co-authors Roman Andriushchenko, Alexander Bork, Milan Češka, Sebastian Junges and Joost-Pieter Katoen for their help with the publication of this work. In the following paragraph, I tried to summarize my contribution to this publication:

I significantly contributed to the formulation of the research ideas, namely the proposed improvements of the inductive synthesis by using different POMDP policies and the whole symbiotic loop. I designed and implemented the methods for obtaining information from belief policies, for using this information to enhance inductive synthesis in Paynt and, for the analysis of obtained FSCs. I came up with the main design of the symbiotic algorithm (see Algorithm 1) and was the one implementing it in our tools. I helped with the implementation of exporting belief policies as well as FSCs found in the inductive synthesis. I performed all of the experiments and pointed out the interesting facts that we present in our evaluation of the proposed ideas. I also took an important part in the process of writing the article.

Structure of this thesis

Chapter 2 introduces the important theory behind the analysis of POMDPs and contains the formulation of the offline synthesis problem for POMDPs this work focuses on. Chapter 3 showcases the current state-of-the-art methods for analysing POMDPs. We focus on introducing methods capable of working with indefinite-horizon specifications, i.e. the belief-based methods with cut-off approximations and the inductive synthesis of FSCs. In Chapter 4 we explore the limitations of the state-of-the-art methods. We showcase the advantage of representing strategies in the form of FSCs and the importance of improving the formal methods in the POMDP synthesis problem. We also highlight the limitations of inductive synthesis and belief-based methods on two simple POMDPs and provide motivation behind the symbiotic integration. Chapter 5 introduces novel ideas for combining belief-based and inductive synthesis approaches. We introduce a framework combining tools Storm and Paynt. And finally, introduce our proposed push-button anytime symbiotic algorithm for the synthesis of FSCs in POMDPs. Chapter 6 contains the experimental evaluation of the implemented integration that shows many practical improvements over the current state-of-the-art tools. And finally, Chapter 7 contains a conclusion of this work.

Chapter 2

Preliminaries and Problem Formulation

This chapter serves as an introduction to the fundamental concepts and tools that are crucial for comprehending the ideas presented in this work. We begin this chapter with an introduction to probability distribution and the basic stochastic models. The chapter then shifts its focus to the specific problem that this work aims to address, which involves synthesizing schedulers for partially-observable Markov decision processes (POMDPs) with respect to indefinite-horizon specifications. POMDPs are critical models for decision-making under uncertainty and limited observability, used in planning autonomous agents, solving games with imperfect information, and medical treatment strategies. The chapter explains the specifications that the work concentrates on and introduces the vital concept of a finite-state controller (FSC), which provides a compact way to represent POMDP policies. Finally, the problem statement for this work is presented.

2.1 Preliminaries

We begin with an explanation of probability distributions, which facilitate reasoning about uncertainty. The Markov property is then introduced as a vital assumption underlying the stochastic decision-making models in this work, derived from the notion of the Markov chain. The chapter proceeds to define Markov decision processes (MDPs), which are an essential model for reasoning about non-deterministic choices in stochastic environments. Furthermore, key concepts surrounding MDPs, such as schedulers, memoryless schedulers, and induced Markov chains, are discussed.

2.1.1 Probability and Markov Property

Probabilistic models are a powerful tool to help us reason about uncertainty, which occurs in many complex systems. A probability distribution describes the likelihood of uncertain outcomes. Formally:

Definition 1 (Probability distribution) *A (discrete) probability distribution over a countable set A is a function $\mu : A \to [0, 1]$ such that the sum $\sum_{a \in A} \mu(a) = 1$. Let $Distr(A)$ denote the set of all probability distributions on A. We define the support of a distribution μ as $supp(\mu) = \{a \in A \mid \mu(a) > 0\}$.*

Example 1 *Let* $A = \{a_0, a_1, a_2\}$ *and let* $\mu : A \to [0,1]$ *be defined as* $\mu : [a_0 \mapsto \frac{1}{3}, a_1 \mapsto \frac{2}{3}, a_2 \mapsto 0]$, $\mu \in Distr(A)$, *i.e.* μ *is a probability distribution on* A. *The support of* μ *is* $supp(\mu) = \{a_0, a_1\}$.

Definition 2 (Markov chain) *A discrete-time Markov chain (MC) is a tuple* $D = (S, s_0, P)$, *where* S *is a finite set of states,* $s_0 \in S$ *is an initial state and* $P : S \to Distr(S)$ *is a transition probability matrix.*

One key property of stochastic models that is often considered is the Markov property. Markov property refers to the memoryless property of stochastic processes. Markov property states that if the current state is known, then the future states of the system are independent of its past states. This allows us to adopt a state-based view of the stochastic models.

Definition 3 (Markov property) *Let* D *be an MC. Let* $X(k) \in S$ *be a random variable describing the current state of* D *at discrete time* $k \geq 0$. *Markov property is defined as* $\mathbb{P}(X(k) = s_k \mid X(k-1) = s_{k-1}, ..., X(0) = s_0) = \mathbb{P}(X(k) = s_k \mid X(k-1) = s_{k-1})$.

2.1.2 Markov Decision Processes

Markov Decision Processes (MDPs) [25] are a widely used framework for modelling decision-making problems under uncertainty. They introduce a non-deterministic choice into the states in the form of actions that can be chosen. The Markov chain defined in the previous section is, in fact, a special case of MDP where in each state there's only one action that can be chosen. MDPs have become widely used in AI, robotics, control theory etc. as they naturally model examples like a robot moving through a stochastic environment.

Definition 4 (MDP) *Markov decision process (MDP) is a tuple* $M = (S, s_0, Act, \mathbf{P})$, *where* S *is a finite set of states,* $s_0 \in S$ *is an initial state,* Act *is a finite set of actions and* $\mathbf{P} : S \times Act \times S \to [0,1]$ *is a transition probability function where for all* $s \in S$ *and* $\alpha \in Act$, $\sum_{s' \in S} \mathbf{P}(s, \alpha, s') \in \{0, 1\}$.

For each state s we can define set $\mathrm{Act}(s) = \{\alpha \in Act \mid \exists s' \in S, \mathbf{P}(s, \alpha, s') > 0\}$. If $\alpha \notin \mathrm{Act}(s)$ then we say that action α cannot be played from state s. We call a state s absorbing if $\forall \alpha \in \mathrm{Act}(s) : \mathbf{P}(s, \alpha, s) = 1$. A path π in MDP is defined as a non-empty (possibly infinite) alternating sequence of states and actions $s_0\alpha_0 s_1\alpha_1 s_2\alpha_2...$ such that $\forall i \in \mathbb{N}_0 : \mathbf{P}(s_i, \alpha_i, s_{i+1}) > 0$. Let $\mathrm{Paths}^M_{inf}(s)$ be a set of all infinite paths from state s and $\mathrm{Paths}^M_{fin}(s)$ a set of all finite paths from state s in an MDP M. Then $\mathrm{Paths}^M(s) = \mathrm{Paths}^M_{inf}(s) \cup \mathrm{Paths}^M_{fin}(s)$. For a finite path $\pi = s_0\alpha_0 s_1\alpha_1...s_n$, let $\mathrm{last}(\pi) = s_n$ denote the last state on the path. The Markov property allows us to compute the probability of an individual path $\pi = s_0\alpha_0 s_1\alpha_1...s_n$ using the probability matrix: $\mathbb{P}[\pi] = \Pi_{i=0}^{n-1} \mathbf{P}(s_i, \alpha_i, s_{i+1})$.

Schedulers (also known as strategies, policies or adversaries) are used to resolve non-determinism in MDPs. Simply put schedulers choose what action should be taken based on path history. In general, we can define a scheduler as:

Definition 5 (Scheduler) *A scheduler is a mapping* $\sigma : Paths^M_{fin} \to Distr(Act)$ *that for a path* $\hat{\pi}$ *yields a probability distribution over actions with* $supp(\sigma(\hat{\pi})) \subseteq Act(last(\hat{\pi}))$, *where* $Paths^M_{fin}$ *is set of all finite paths in* M.

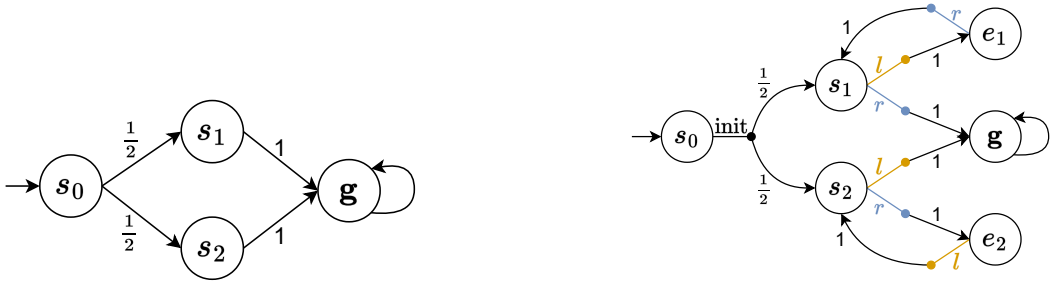
Let Σ^M denote the set of all schedulers for MDP M.

Definition 6 (Memoryless Scheduler) *Scheduler σ is memoryless if for paths $\hat{\pi}$, $\hat{\pi}'$ it holds that: $last(\hat{\pi}) = last(\hat{\pi}') \implies \sigma(\hat{\pi}) = \sigma(\hat{\pi}')$. For memoryless schedulers we often use $\sigma(last(\hat{\pi}))$ instead of $\sigma(\hat{\pi})$.*

In most of the cases in this work, we will only consider deterministic schedulers, i.e. the mapping σ chooses just one action instead of distribution over actions. Formally, σ is deterministic iff $|supp(\sigma(\pi))| = 1$ for all $\pi \in \text{Paths}^M$. We can say that deterministic schedulers are therefore of type $\text{Paths}^M \to Act$. In general, randomized schedulers are more powerful, however, they are harder to interpret and are therefore impractical in most cases. What is really important in the analysis of MDPs is that for most basic specifications we need to only consider memoryless schedulers to find the minimum and maximum value schedulers [25]. This makes the set of considered schedulers finite. By applying a memoryless scheduler to MDP we get an $|S|$-state induced MC (if we needed memory for the scheduler we would need to encode it in the induced model). Formally:

Definition 7 (Induced MC) *Let $M = (S, s_0, Act, \mathbf{P})$ be an MDP. We say that the scheduler $\sigma \in \Sigma^M$ induces an MC $M^\sigma = (Paths^M_{fin}(s_0), s_0, P^\sigma)$ where $P^\sigma(\pi, \pi\sigma(\pi)s') = P(last(\pi), \sigma(\pi), s')$.*

Example 2 *Let's consider MDP $M = (\{s_0, s_1, s_2, e_1, e_2, g\}, s_0, \{init, l, r\}, \mathbf{P})$ where the transition probability function $\mathbf{P}(s_0, init, s_1) = \frac{1}{2}, \mathbf{P}(s_0, init, s_2) = \frac{1}{2}, \mathbf{P}(s_1, l, e_1) = 1, \mathbf{P}(s_1, r, g) = 1, \mathbf{P}(s_2, r, e_2) = 1, \mathbf{P}(s_2, l, g) = 1, \mathbf{P}(e_1, r, s_1) = 1, \mathbf{P}(e_2, l, s_2) = 1$ and state g is absorbing. MDP M is shown in the form of a state transition graph in Figure 2.1b. Let σ be a deterministic memoryless scheduler such that $\sigma(s_1) = r$ and $\sigma(s_2) = l$. If we apply scheduler σ to MDP M we obtain an induced Markov chain that is isomorphic to a four-state MC (shown in Figure 2.1a) $D = (\{s_0, s_1, s_2, g\}, s_0, P)$ where $P(s_0) = [s_1 \mapsto \frac{1}{2}, s_2 \mapsto \frac{1}{2}], P(s_1) = [g \mapsto 1], P(s_2) = [g \mapsto 1], P(g) = [g \mapsto 1]$.*



(a) (b)

Figure 2.1: (a) contains a graph representation of MC with 4 states. State G is an absorbing state (transition probability omitted for clarity). (b) showcases graph representation of MDP with 6 states and 3 actions. MC depicted in (a) is isomorphic with an induced MC obtained by applying a scheduler σ, $\sigma(s_1) = r$ and $\sigma(s_2) = l$, to the MDP depicted in (b).

2.2 Partially-observable Markov Decision Processes

Partially observable Markov decision processes (POMDPs) [24] emerged as a natural extension of MDPs and are currently widely researched. They help us model another level of

uncertainty in the form of observations. These observations restrict the information about the current state. Any two states with the same observation are indistinguishable from each other and we need to remember the history of visited observations to make good choices.

Definition 8 (POMDP) *Partially observable MDP (POMDP) is a tuple* $\mathcal{M} = (M, Z, O)$*, where* $M = (S, s_0, Act, \mathbf{P})$ *is the underlying MDP,* Z *is a finite set of observations and* $O : S \to Z$ *is an observation function*[1]*.*

We can extend the observation function to work on paths. Let $\pi = s_0\alpha_0 s_1\alpha_1 s_2\alpha_2...$ be a path in a POMDP $O'(\pi) = O(s_0)\alpha_0 O(s_1)\alpha_1 O(s_2)\alpha_2...$ in the rest of the paper we will write $O(\pi)$ to represent $O'(\pi)$. Two paths π_1, π_2 with $O(\pi_1) = O(\pi_2)$ are called observation-equivalent and are indistinguishable from the decision-making perspective.

W.l.o.g., we assume that all states with the same observation have the same set of enabled actions. Formally, $\forall s, s' \in S : O(s) = O(s') \implies \text{Act}(s) = \text{Act}(s')$. This means that if we consider state s with observation z we can write $\text{Act}(z) = \text{Act}(s)$.

Schedulers again help us to resolve POMDPs, however, finding the optimal scheduler even for simple specifications without a discounting factor or a finite step bound is undecidable in general. This comes from the fact that the optimal scheduler may require infinite memory. In contrast with MDPs, in POMDPs we have to consider a special case of schedulers called observation-based schedulers:

Definition 9 (Observation-based Scheduler) *Scheduler* σ *is observation-based if for paths* $\hat{\pi}$*,* $\hat{\pi}'$ *it holds that:* $O(\hat{\pi}) = O(\hat{\pi}') \implies \sigma(\hat{\pi}) = \sigma(\hat{\pi}')$

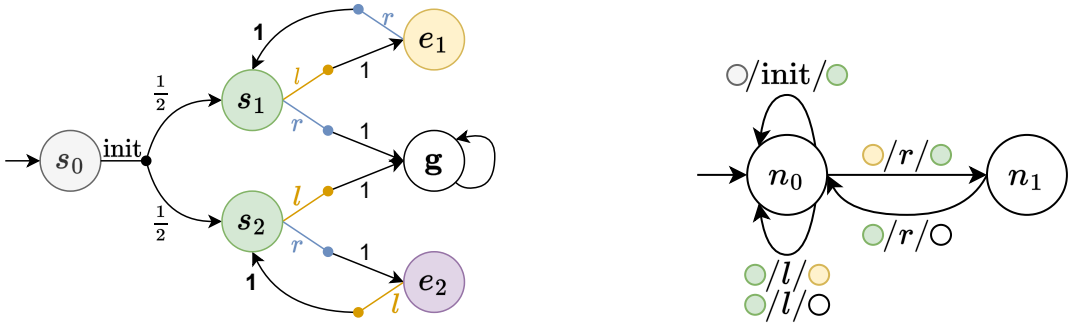
Observation-based schedulers make a decision based on the history of observations instead of states as is the case in MDPs. Let $\Sigma^{\mathcal{M}}_{obs}$ denote the set of all observation-based schedulers for a POMDP $\mathcal{M}$. If we take a scheduler $\sigma \in \Sigma^{\mathcal{M}}_{obs}$ and apply it to POMDP $\mathcal{M}$ we induce an MC $\mathcal{M}^\sigma$ in a similar fashion as in MDPs.

Example 3 *Let's consider a POMDP* $\mathcal{M} = (M, \{in, mid, left, right, goal\}, O)$ *where* M *is an MDP from Example 2 and the observation function is defined in the following way* $O = \{s_0 \mapsto in, s_1 \mapsto mid, s_2 \mapsto mid, e_1 \mapsto left, e_2 \mapsto right, g \mapsto goal\}$*. The graph representation for this POMDP is shown in Figure 2.2a. The graph representation is similar to the MDP graph representation with the exception that we colour states in the graph to signify what is the observation of each state. In Figure 2.2a we assigned the colours in the following way* $in \mapsto grey, mid \mapsto green, left \mapsto yellow, right \mapsto purple$ *and* $goal \mapsto white$*.*

2.3 Specification

In this work, we consider indefinite-horizon reachability or expected total reward properties. Formally, let's consider MC $D = (S, s_0, \mathbf{P})$ and let $T \subseteq S$ be a set of target states. $\mathbb{P}^D[s \models \Diamond T]$ denotes the probability of reaching T from state $s \in S$. If we use $\mathbb{P}^D[\Diamond T]$ we mean $\mathbb{P}^D[s_0 \models \Diamond T]$, i.e. the probability of reaching T from the initial state and we omit the superscript if the MC is clear from the context. Now assume POMDP $\mathcal{M}$ with underlying MDP $M = (S, Act, \mathbf{P}, s_0)$ and a set $T \subseteq S$ of absorbing target states. W.l.o.g. we assume that there is a special observation $z^T \in Z$ assigned to each of the target states,

[1] We can encode more general observation functions using this formalism



(a) (b)

Figure 2.2: (a) contains an example of a graph representation of a POMDP. The underlying MDP is taken from Figure 2.1b and is extended with 5 observations (represented by 5 colours). (b) contains a simple 2-state FSC. Each transition is representing a triple prior observation, chosen action and posterior observation. Transitions that are not feasible for the POMDP from (a) were omitted. This showcases the fact that the general FSC as defined in Section 2.4 is not practical in many cases. The FSC from (b) represents the best observation-based strategy if we want to minimize the number of steps it takes to reach state G in POMDP from (a).

i.e. $\forall s \in S : s \in T \iff O(s) = z^T$. The maximal reachability probability of T from state $s \in S$ in $\mathcal{M}$ is $\mathbb{P}^{\mathcal{M}}_{max}[s \models \Diamond T] := \sup_{\sigma \in \Sigma_{obs}} \mathbb{P}^{\mathcal{M}^\sigma}[s \models \Diamond T]$. The minimal reachability probability $\mathbb{P}^{\mathcal{M}}_{min}[s \models \Diamond T]$ is defined analogously. These specifications are used for long-term planning and formal verification.

Some methods for POMDP analysis consider infinite-horizon specifications. These specifications still consider an unbounded number of decisions, however, they introduce a discount factor γ between 0 and 1. This discount factor means that actions in the present are more significant than the actions considered in the future. The closer the discount factor gets to 1, the more we need to consider future decisions. Another possibility is to only consider bounded reachability or rewards. This means that you only analyse the model with respect to some number of bounded actions. These specifications are also known as finite-horizon specifications.

2.4 Finite-state Controllers

Finite-state controllers [2] are mealy automata that encode schedulers in a compact way. They also have the advantage of being easy to use and verify.

Definition 10 (FSC) *Finite-state controller (FSC) for POMDP $\mathcal{M}$ is a tuple $F = (N, n_0, \gamma, \delta)$, where N is a finite set of memory nodes, $n_0 \in N$ is the initial memory node, $\gamma : N \times Z \to Act$ is the action mapping function and $\delta : N \times Z \times Z \to N$ is the memory update function.*

FSCs represent observation-based schedulers for POMDPs and therefore can be used to induce MC. We call the states of FSC memory nodes (or just nodes) to distinguish them from POMDP states. The action mapping function selects the actions based on the current node of the FSC and the current observation, while the memory update function looks

at the current node, current observation and the next observation seen after playing the action chosen by the action mapping function. So for example for POMDP in state s with observation $z = O(s)$ an agent following the strategy defined by FSC F executes action $\alpha = \gamma(n, z)$ associated with the current memory node of the FSC and the current (prior) observation z. The POMDP state is updated to s' with $\mathbf{P}(s, \alpha, s') > 0$ and based on the next (posterior) observation $z' = O(s')$, the FSC updates its memory node to $n' = \delta(n, z, z')$. In this work, we only consider deterministic FSCs, however, it is possible to define FSCs whose action mapping function returns a distribution over actions and the memory update function a distribution over memory nodes in general. For $|N| = k$ we call an FSC a k-FSC. If $k = 1$, the FSC encodes a memoryless policy. The induced MC for FSC F and POMDP $\mathcal{M}$ is $\mathcal{M}^F = (S \times N, (s_0, n_0), \mathbf{P}^F)$ where for all $(s, n), (s', n') \in S \times N$ we define

$$\mathbf{P}^F((s, n), (s', n')) = [n' = \delta(n, O(s), O(s'))] \cdot \mathbf{P}(s, \gamma(n, O(s)), s')$$

Example 4 *Let's consider POMDP $\mathcal{M}$ from Example 3. Let FSC $F = (\{n_0, n_1\}, n_0, \gamma, \delta)$, where $\gamma = \{(n_0, in) \mapsto init, (n_0, mid) \mapsto l, (n_0, left) \mapsto r, (n_1, mid) \mapsto r\}$ and $\gamma = \{(n_0, in, mid) \mapsto n_0, (n_0, mid, left) \mapsto n_0, (n_0, mid, goal) \mapsto n_0, (n_0, left, mid) \mapsto n_1, (n_1, mid, goal) \mapsto n_0\}$. The state diagram for this FSC is shown in Figure 2.2b.*

Definition 11 (Family of FSCs) *A family of k-FSCs for POMDP $\mathcal{M}$ is a tuple $\mathcal{F}_k^{\mathcal{M}} = (N, n_0, K)$, where N is the set of k nodes, $n_0 \in N$ is the initial node and $K = N \times Z \times Z$ is a finite set of parameters each with domain $V_{(n,z,z')} \subseteq Act \times N$.*

We can obtain an FSC from a family by choosing the value for each parameter. Families of FSCs contain $\mathcal{O}((|Act| \cdot |N|)^{(|N| \cdot |Z| \cdot |Z|)})$ many FSCs. A POMDP and a family of FSCs induce a family of Markov chains. We use $\mathcal{F}^{\mathcal{M}}$ to denote the family of all FSCs for POMDP $\mathcal{M}$.

FSCs with k nodes provide the same amount of memory for every observation, however, in many practical problems memory is often required only for some of the observations. Therefore we can consider reduced FSCs given by a memory model $\mu : Z \to \mathbb{N}$, where $\mu(z)$ determines the number of memory nodes used in the observation z. This way the number of nodes in the FSC can remain the same, but the parameter domains can be significantly reduced. The family of reduced FSCs induces a smaller design space. The definition 10 introduces a general form of an FSC also called posterior-aware FSC. In many cases, we only need to consider a subset of FSCs called posterior-unaware FSCs. We say an FSC with update function δ is posterior-unaware if the posterior observations are not taken into consideration when updating the memory, i.e. $\delta(n, z, z') = \delta(n, z, z'')$ for all $n \in N$ and for all $z, z', z'' \in Z$. This restriction reduces the number of FSCs in the families we consider (upper bound on the number of FSCs $\mathcal{O}((|Act| \cdot |N|)^{(|N| \cdot |Z|)})$). The advantage of the general FSCs is that they usually need fewer memory nodes to encode the same strategy. We will get to a proper definition of the so-called μ-FSC and belief FSC and overview posterior aware and unaware FSCs in the Section 5.3.

For MDPs with infinite state space and general POMDPs, an FSC realising the maximal (minimal) reachability probability generally does not exist. Let's take FSC $F \in \mathcal{F}^{\mathcal{M}}$ with memory nodes N. Let $\mathbb{P}^{\mathcal{M}^F}[(s, n) \models \Diamond T] := \mathbb{P}^{\mathcal{M}^F}[(s, n) \models \Diamond(T \times N)]$ denote the probability of reaching target states T from state $(s, n) \in S \times N$. And similarly, $\mathbb{P}^{\mathcal{M}^F}[\Diamond T] := \mathbb{P}^{\mathcal{M}^F}[\Diamond(T \times N)]$ denotes the probability of reaching target states T in the MC $\mathcal{M}^F$ induced by F on $\mathcal{M}$.

2.5 Problem Statement

We focus on the indefinite-horizon specifications in this work. We, therefore, focus on long-term goals. For these specifications, finding the optimal FSC is an undecidable problem. The classical synthesis problem [23] for POMDPs asks: given a POMDP $\mathcal{M}$, set of target states T and a threshold λ, find an FSC F for which it holds that $\mathbb{P}^{\mathcal{M}^F}[\Diamond T] \geq \lambda$ if such an FSC exists. We focus on a more practical take and aim to optimise the value $\mathbb{P}^{\mathcal{M}^F}[\Diamond T]$ in an anytime fashion: we try to find FSCs with high value and the faster we are able to do it, the better.

Other variants of the synthesis problem e.g. maximising synthesis problem for the expected total reward and minimisation variants are defined analogously. For simplicity, in this work, we will assume that we want to always maximise the value. Of course, the proposed method works for the minimisation problem as well, which is showcased in the chapter with experimental evaluation.

The value of the FSC F is the main objective of our problem statement, however, in addition to that, we also look at the size of the FSC as a secondary objective and discuss it in detail in later chapters.

Chapter 3

State-of-the-art Methods

In this chapter, we will review the state-of-the-art methods for the analysis of POMDPs. This work focuses on the offline synthesis problem for indefinite-horizon specifications as was highlighted in the previous chapter. For this reason, we will mainly focus on introducing belief-based methods with cut-off approximations and the inductive synthesis of FSCs. Belief-based methods are a class of methods that operate on probability distributions over the state space. These distributions are also known as beliefs. Inductive synthesis aims to find compact FSCs representing good policies. It constructs families of FSCs and tries to find the best FSC in the given family using the results it computed up to that point. We also introduce the point-based methods which perform well when a discount factor is introduced, but struggle in long-term planning. At the end of the chapter, we also review the simulation-based approach and the reinforcement learning methods. These methods are strong when we work with prohibitively large or unknown POMDPs and they are typically used in online planning, however, they tend to be data-intensive, the efficiency of sampling limits their performance and their results are difficult to interpret and verify compared to the FSCs we consider in the most of this work.

3.1 Belief-based Methods

This section introduces the methods for POMDP analysis that build on the notion of a belief. We introduce this very important notion and all the needed theories using it. After that, we introduce more concepts that are used in state-of-the-art belief-based methods.

3.1.1 Beliefs and Belief MDP

One way of analysing POMDPs is to construct a so-called (fully observable) belief MDP and perform the analysis on this MDP [28]. The state space of this MDP consists of beliefs: probability distributions over states of the POMDP $\mathcal{M}$ having the same observation. The notion of belief is therefore really important and it lets us to describe the uncertainty (or partial observability) of the current state. Let $S_z := \{s \in S \mid O(s) = z\}$ denote the set of all states in POMDP $\mathcal{M}$ having the same observation $z \in Z$. Let the set of all possible beliefs be $\mathcal{B}_{\mathcal{M}} := \bigcup_{z \in Z} Distr(S_z)$. Notice that the set of all possible beliefs is uncountable. For any belief $b \in \mathcal{B}_{\mathcal{M}}$ by $O(b) \in Z$ we denote the unique observation $O(s)$ of any $s \in supp(b)$.

Let $\mathbf{P}(s,\alpha,z) = \sum_{s' \in S}[O(s') = z] \cdot \mathbf{P}(s,\alpha,s')$ denote[1] the probability to move to a state with observation z from state s using action α and let $\mathbf{P}(b,\alpha,z) = \sum_{s \in S} b(s) \cdot \mathbf{P}(s,\alpha,z)$ denote the probability to observe z after taking action α in belief b. We now can define the belief obtained by taking action α from belief b conditioned on observing z if $\mathbf{P}(b,\alpha,z) > 0$ as:

$$\langle b \mid \alpha, z \rangle(s') = \frac{[O(s') = z] \cdot \sum_{s \in S} b(s) \cdot \mathbf{P}(s,\alpha,s')}{\mathbf{P}(b,\alpha,z)}$$

for all $s' \in S_{z'}$. If $\mathbf{P}(b,\alpha,z) = 0$ the next belief is undefined.

Definition 12 (Belief MDP) *The belief MDP of POMDP $\mathcal{M} = (M, Z, O)$ is the MDP $\mathcal{M}^{\mathcal{B}} = (\mathcal{B}_{\mathcal{M}}, b_0, Act, \mathbf{P}^{\mathcal{B}})$, where $\mathcal{B}_{\mathcal{M}}$ are the beliefs representing the states, $b_0 = \{s_0 \mapsto 1\}$ is the initial belief, and transition function $\mathbf{P}^{\mathcal{B}}$:*

$$\mathbf{P}^{\mathcal{B}}(b,\alpha,b') = \begin{cases} \mathbf{P}(b,\alpha,O(b')) & \text{if } b' = \langle b \mid \alpha, O(b') \rangle \\ 0 & \text{otherwise} \end{cases}$$

What is really important about belief MDPs is the fact that they accurately capture the behaviour in their corresponding POMDP. By analysing belief MDP we are able to get the scheduler for the POMDP directly. The analysis can be done using the standard MDP model checking techniques.

The problem that arises in the construction of belief MDPs is the fact that their reachable state space might be infinite. And this problem shows itself even for very small POMDPs as will be shown in later chapters. If this is the case then we have to only construct some finite part of the belief MDP and approximate the rest using appropriate approximation techniques.

Example 5 *Let's consider POMDP $\mathcal{M}$ from Example 3. For this POMDP we can construct a finite belief MDP $\mathcal{M}^{\mathcal{B}} = (\{b_0, b_1, b_2, b_3, b_4, b_5, b_g\}, b_0, \{init, l, r\}, \mathbf{P}^{\mathcal{B}})$ where $\mathbf{P}^{\mathcal{B}} = \{(b_0, init, b_1) \mapsto 1, (b_1, l, b_2) \mapsto \frac{1}{2}, (b_1, l, b_g) \mapsto \frac{1}{2}, (b_1, r, b_3) \mapsto \frac{1}{2}, (b_1, r, b_g) \mapsto \frac{1}{2}, (b_2, r, b_4) \mapsto 1, (b_3, l, b_5) \mapsto 1, (b_4, l, b_2) \mapsto 1, (b_4, r, b_g) \mapsto 1, (b_5, r, b_3) \mapsto 1, (b_5, l, b_g) \mapsto 1\}$. The beliefs encode following distributions over states: $b_0 = [s_0 \mapsto 1], b_1 = [s_1 \mapsto \frac{1}{2}, s_2 \mapsto \frac{1}{2}], b_2 = [e_1 \mapsto 1], b_3 = [e_2 \mapsto 1], b_4 = [s_1 \mapsto 1], b_5 = [s_2 \mapsto 1], b_g = [g \mapsto 1]$. This belief MDP is shown in the form of a state transition graph in Figure 3.1.*

3.1.2 Belief Exploration

We start the exploration of the belief space from the initial belief $b_0 = \{s_0 \mapsto 1\}$. Since all states in a given belief have the same observation we can get the set of actions for belief b as $\mathrm{Act}(b) = \mathrm{Act}(s)$ for any $s \in supp(b)$. In the exploration, we want to expand all of the actions of the current belief and get a set of new beliefs which are put in a queue. Each time a belief is picked from the queue, there is a decision on whether it should be expanded or left unexplored. Assuming $\mathcal{M}^{\mathcal{B}}$ is unfolded up to some depth, let $\mathcal{E} \subset \mathcal{B}_{\mathcal{M}}$ denote the set of explored beliefs and let $\mathcal{U} \subset \mathcal{B}_{\mathcal{M}} \backslash \mathcal{E}$ denote the frontier. Frontier is the set of unexplored beliefs reachable from $\mathcal{E}$ in one step. The idea of explored belief space $\mathcal{E}$ and of the frontier $\mathcal{U}$ is shown in Figure 3.2a. If we decide to expand a state from the frontier further we compute the set of new beliefs for each action using the formula presented in Section 3.1.1.

[1] Iverson bracket notation: [x] = 1 if x is true and 0 otherwise

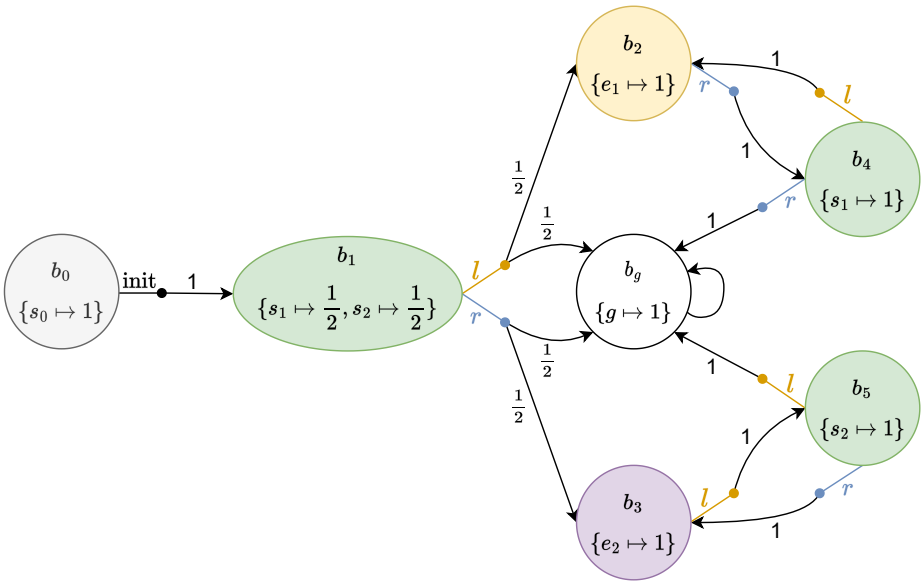


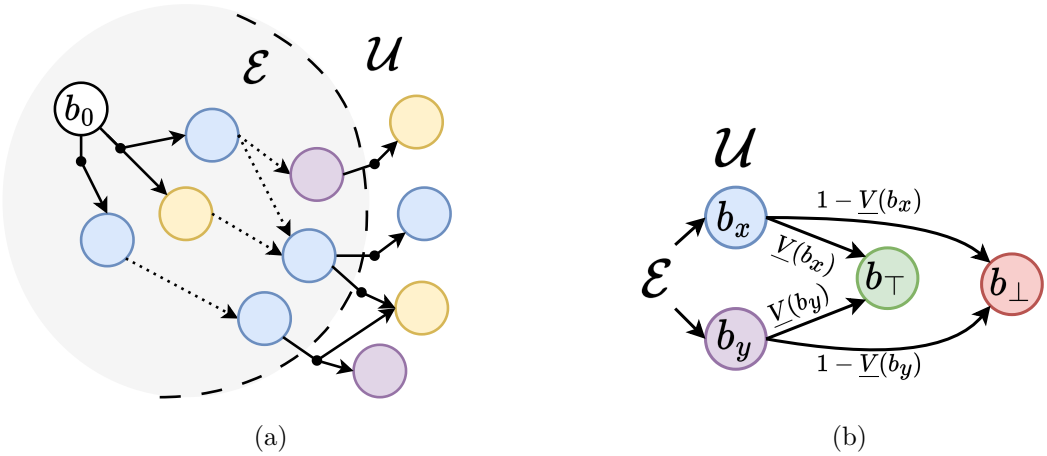
Figure 3.1: Belief MDP constructed for POMDP from Figure 2.2a. For every belief there is a corresponding distribution over original states. This belief MDP is complete, i.e. the reachable belief space for the considered POMDP is finite. The colours are used only to help us map the beliefs to the observations of the original POMDP but there are no observations in the belief MDP itself.

Note that if we are expanding belief b on action α obtaining a set of new beliefs $\mathcal{B}^{new}_{\mathcal{M}}$ then for each observation z the sum $\sum_{b \in \mathcal{B}^{new}_{\mathcal{M}}} [O(b) = z] \leq 1$ which means that no two beliefs from the set $\mathcal{B}^{new}_{\mathcal{M}}$ have the same observation. To construct belief MDP we add transitions from belief b on action α to the new beliefs with their corresponding computed probabilities. In the set of new beliefs, we might get a belief b' which we explored before meaning that we add a transition from b to already existing b' in the belief MDP. If we choose to not expand a belief from the frontier then we have to approximate its value which is discussed in the next section. One way to decide what beliefs to explore is to set a depth limit or to have a heuristic which takes the gap between the fully observable lower bound and upper bound into consideration. We usually work with a combination of these, i.e. we set a size for the belief MDP but we want to focus on exploring paths that are important for the overall result. If there are no beliefs in the frontier, the whole reachable belief space has been explored so we have constructed a finite belief MDP and we can easily model check it without any approximations (an example of a finite belief MDP is shown in the Example 5).

Approximating the values of unexplored beliefs is an essential part of the belief-based methods. There are a lot of models where finite exploration becomes ineffective after a number of beliefs were explored and the overall result therefore heavily depends on the approximations. Therefore, there is a lot of focus on coming up with good approximations.

3.1.3 Cut-off Approximation of Unexplored Belief Space

The main idea behind belief cut-offs is that we want to assume a target state is reached immediately from the frontier (or cut-off) state while achieving sub-optimal value. This idea can be used for computing both over-approximations described in [6] and under-approximations described in [7]. In this work, we will focus on under-approximations as we



(a) (b)

Figure 3.2: (a) showcases the explored part of the belief space $\mathcal{E}$ from initial belief b_0 and the frontier of the belief MDP $\mathcal{U}$ reachable in one step from $\mathcal{E}$. (b) Application of value function $\underline{V}$ to compute cut-off values for each belief $b \in \mathcal{U}$ and obtain a belief MDP abstraction $\overline{\mathcal{M}^{\mathcal{B}}}$. The states $b_\top$ and $b_\perp$ are newly added sink states, where the former is added to the set of target states. Note that beliefs $b \in \mathcal{U}$ can have multiple actions but we omitted them for clarity.

are interested in finding real usable controllers. Let's say our specification is to maximize the probability of reaching a target state. One simple way how to approximate cut-off states, in this case, is to make all of them absorbing and therefore there is no path to the target states from cut-off states. This gives us a proper under-approximation of the real value. Of course, we can do better than that by searching for strategies for the unexplored part of the belief space beyond the cut-off states. Applying cut-offs directly manipulates the belief MDP and yields a finite model.

Let $\underline{V} : \mathcal{B}_{\mathcal{M}} \to [0,1]$ be a function where $\forall b \in \mathcal{B}_{\mathcal{M}} : \underline{V}(b) \leq V^*(b)$. We call $\underline{V}$ and under-approximative value function and $\underline{V}(b)$ the cut-off value of belief b. The under-approximative function can be defined for maximizing/minimizing rewards as well. By applying this function to all unexplored beliefs from the frontier we not only under-approximate their values but we also under-approximate values of all the explored beliefs in the belief MDP. With this we obtain a finite approximation $\overline{\mathcal{M}^{\mathcal{B}}}$. Figure 3.2b shows an example of applying function $\underline{V}$ to state in the frontier. The question of finding an adequate under-approximative function $\underline{V}$ is crucial for the cut-off approach. One simple approach was explained in the previous paragraph but if we want a better approximation a more sophisticated approach is needed. One such approach is to assign some arbitrary fixed observation policy $\sigma \in \Sigma^{\mathcal{M}}_{obs}$. Let $U^\sigma : S \to [0,1]$ such that for all $s \in S$ we have $U^\sigma(s) = \mathbb{P}^{\mathcal{M}^\sigma}[s \models \Diamond T]$. Then we can define the function $\mathcal{U}^\sigma : \mathcal{B}_{\mathcal{M}} \to [0,1]$ as $\mathcal{U}^\sigma(b) := \sum_{s \in supp(b)} b(s) \cdot U^\sigma(s)$.

Lemma 1 *[7] The function $\mathcal{U}^\sigma$ is an under-approximative value function, i.e. for all $b \in \mathcal{B}_{\mathcal{M}}$:*

$$\mathcal{U}^\sigma(b) := \sum_{s \in supp(b)} b(s) \cdot U^\sigma(s) \leq V^*(b).$$

By Lemma 1 we know that finding an adequate under-approximative value function reduces to finding „good“ observation-based controllers for $\mathcal{M}$. We will use this fact to enhance the

belief-based algorithm proposed in [7], where function $\mathcal{U}^\sigma$ is given by heuristic which uses values obtained from the fully observable underlying MDP.

Belief clipping

An alternative method for approximating unexplored beliefs called belief clipping was introduced in [7], however, in the experiments we showcase that this method brings practical benefits only for a very limited selection of models. Therefore we will not explain this method in detail in this work and only give the high-level ideas behind it. The intuition behind belief clipping is that we shift some of the probability mass of a belief b in order to transform b to a new belief $\widetilde{b}$. We then want to connect b with $\widetilde{b}$ in a way that the accuracy of approximation of the value $V^*(b)$ depends only on the approximation of $V^*(\widetilde{b})$ and a so-called clipping value which represents a notion of distance between beliefs b and $\widetilde{b}$. We can then explore the successors of $\widetilde{b}$ to obtain good approximations for both b and $\widetilde{b}$.

3.1.4 Point-based Approximations

Point-based approximations work by discretizing the belief space into a set of representative belief points. These points are used to approximate the optimal value function and policy. The algorithm iteratively updates the value function and policy at each point until convergence. The belief points are chosen based on their ability to improve the approximation of the value function and policy. This approach reduces the computational complexity of solving POMDPs by focusing on a smaller set of belief points rather than the entire belief space. The most notable point-based algorithm is SARSOP [21]. SARSOP uses a tree-based data structure to efficiently search the near-optimal belief space and select the most promising actions.

3.2 Inductive Synthesis of FSCs

In this section, we recap a recent inductive approach [4] for finding FSCs for POMDPs that builds on inductive methods for synthesis of probabilistic programs [3] First, we showcase the overall synthesis framework for POMDPs, and then we introduce abstraction refinement as one of the main methods for the exploration of the family of FSCs, lastly, we discuss the importance of memory injection in inductive synthesis.

3.2.1 Synthesis Framework

The inductive synthesis framework works in two stages, the inner stage and the outer stage. The overall framework is showcased in Figure 3.3. Let's first discuss the outer stage. The main part of the outer stage is called the learner. Learner constructs a family of FSCs also called the design space. From this design space, a part called teacher provides the best FSC and potentially some additional information about the result. The learner receives the candidate FSC and either accepts it as the final result or adapts the design space and the loop continues. Naturally, the teacher will provide results faster when the design space is small, however, when constructing small design spaces we have to be careful not to prune potentially good FSCs. So the main objective of the outer loop is to start with a small design space and strategically adapt it based on the results from the teacher.

The internals of the Teacher is called the inner stage of the synthesis. Its task is to determine the best FSC within the given design space. A naive implementation of the

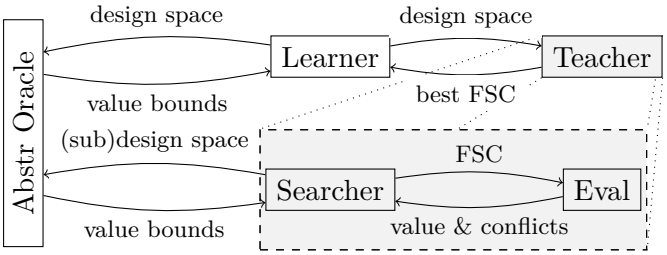


Figure 3.3: Nested inductive synthesis framework with an abstraction oracle. The framework takes a POMDP and a specification and finds an FSC that satisfies the specification. The learner is responsible for creating a design space of candidate FSCs. The teacher finds the best FSC from the suggested family of FSCs. This Figure is taken from [4].

teacher enumerates all FSCs in the design space. The inductive synthesis we consider in this work realizes the teacher through another inductive synthesis loop. It uses abstraction refinement (see Section 3.2.2) to search the design space effectively. Both learning stages have access to an oracle that over-approximates the design space. This larger abstract design space can be analysed efficiently as the underlying problem. This resembles the analysis of fully observable schedulers. Therefore, the oracle yields a constraint on what the best FSC from the original design space can achieve. This information is essential for guiding the search in both stages.

We assume an inductive synthesis for finding deterministic posterior-unaware FSCs in this work. Deterministic FSCs are beneficial in terms of the reproducibility of their behaviour, which makes working with them easier and more practical. Moreover, finding randomised FSCs is a more complex problem[2].

3.2.2 Abstraction Refinement

Consider a finite family of FSCs $\mathcal{F}_k^{\mathcal{M}}$ of k-FSCs with memory nodes $N = \{n_0, n_1, ..., n_{k-1}\}$ and the associated family $\mathcal{M}^{\mathcal{F}_k^{\mathcal{M}}} := \{\mathcal{M}^F \mid F \in \mathcal{F}_k^{\mathcal{M}}\}$ of induced MCs. The states of each MC are tuples $(s, n) \in S \times N$. We can create an MDP abstraction of the family $\mathcal{M}^{\mathcal{F}_k^{\mathcal{M}}}$ of MCs to help us reason about the whole family. Informally, the MDP abstraction of the family $\mathcal{M}^{\mathcal{F}_k^{\mathcal{M}}}$ is an MDP $MDP(\mathcal{F}_k^{\mathcal{M}})$ with the set $S \times N$ of states such that, if some MC $M \in \mathcal{M}^{\mathcal{F}_k^{\mathcal{M}}}$ executes action α[3] in state $(s, n) \in S \times N$, then this action is also enabled in the state (s, n) of $MDP(\mathcal{F}_k^{\mathcal{M}})$ with the same effect. This means that $MDP(\mathcal{F}_k^{\mathcal{M}})$ over-approximates the behaviour of all the MCs in the family $\mathcal{M}^{\mathcal{F}_k^{\mathcal{M}}}$ (all FSCs in the family $\mathcal{F}_k^{\mathcal{M}}$). It is an over-approximation as in every step an arbitrary family member is simulated and it may switch between steps. MDP abstraction formally:

Definition 13 (MDP abstraction) *MDP abstraction for POMDP $\mathcal{M}$ and family $\mathcal{F}_k^{\mathcal{M}} = \{F_1, F_2, ...F_m\}$ of k-FSCs is the MDP $MDP(\mathcal{F}_k^{\mathcal{M}}) := (S \times N, (s_0, n_0), \{1, 2, ..., m\}, \mathbf{P}^{\mathcal{F}_k^{\mathcal{M}}})$ where*

$$\mathbf{P}^{\mathcal{F}_k^{\mathcal{M}}}((s, n), i) = \mathbf{P}^{F^i}$$

Even though this MDP has m actions, practically, many actions may coincide. Let's observe that this abstract MDP definition represents a proper abstraction:

[2]Finding randomised FSCs is ETR-complete, meanwhile finding deterministic FSCs is NP-complete [20].

[3]In our formal definition of a MC we do not include actions, however, you can think of MC as an MDP where each state has one action enabled.

Lemma 2 *[10] For all* $F \in \mathcal{F}_k^{\mathcal{M}}$, $\mathbb{P}_{\min}^{MDP(\mathcal{F}_k^{\mathcal{M}})}[\Diamond T] \leq \mathbb{P}^{\mathcal{M}^F}[\Diamond T] \leq \mathbb{P}_{\max}^{MDP(\mathcal{F}_k^{\mathcal{M}})}[\Diamond T]$.

By this lemma, we know that the analysis of the abstraction MDP gives us bounds on the best FSC from the family $\mathcal{F}_k^{\mathcal{M}}$. This fact is an important part of abstraction refinement. Because the abstraction MDP over-approximates the behaviour of FSCs in family $\mathcal{F}_k^{\mathcal{M}}$, not every scheduler σ induces an FSC. We say that a scheduler σ is consistent if $O(s) = O(s') \implies \sigma((s,n)) = \sigma((s',n))$ for all $s, s' \in S$ and $n \in N$. The set of consistent schedulers in $MDP(\mathcal{F}_k^{\mathcal{M}})$ corresponds to the family $\mathcal{F}_k^{\mathcal{M}}$.

The main idea behind abstraction refinement is that if the best policy in the abstraction MDP is worse than the given bound then all the FSCs in the original family violate the bound. Assume we want to maximise the constraint $\mathbb{P}_{\geq\lambda}[\Diamond T]$ for some target set T. Let the scheduler σ^* be the optimal scheduler for the given constraint in our MDP $MDP(\mathcal{F})$ and let its achieved probability be $\mathbb{P}^{\sigma^*}$. If $\mathbb{P}^{\sigma^*} < \lambda$ then it holds that all $F \in \mathcal{F}$ violate the constraint and the whole family can be pruned. Otherwise, if σ^* is a consistent scheduler then it represents a valid FSC satisfying our constraint which is what we wanted to find. The last possibility is that the analysis of $MDP(\mathcal{F})$ is inconclusive and the family $\mathcal{F}$ needs to be refined. Additionally, this analysis also provides bounds that are used in both the inner and the outer synthesis loops.

The refinement strategy drives the exploration of family $\mathcal{F}_k^{\mathcal{M}}$. It decomposes a given family into smaller subfamilies by splitting the domain of selected parameters from K. The key idea is to examine the inconsistencies of the scheduler σ^*. The refinement strategy estimates the significance of each inconsistent parameter $p \in K$ for σ^* by examining the impact of changing p and this impact is weighted by the expected occurrence frequency of the decisions corresponding to p. The most significant parameter p is selected. Assume (inconsistent) p with domain $V_p = \{v_1, v_2, ..., v_n\}$ and scheduler σ^* selected options v_i and v_j. The refinement partitions V_p into three subdomains $V_p^1 = \{v_i\}$, $V_p^2 = \{v_j\}$ and $V_p^3 = V_p \backslash \{v_i, v_j\}$ corresponding to three new subfamilies. This removes the inconsistency and the process of analysing families continues until the best FSC from the original family is found.

3.2.3 Memory Injection Strategy

Memory injection strategy dictates how the outer stage of the synthesis constructs the design space. The design space represents a subset of FSCs and is passed to the teacher for analysis. This construction assumes access to the bounds of the abstraction presented in the previous section as is outlined in the scheme from Figure 3.3. The learner processes this information and derives a new design space. It does this by adding memory. By adding memory we mean considering FSCs with more states, i.e. the FSCs can store more information. This helps FSCs to represent better strategies, however, it (dramatically) increases the size of the design space which is one of the big challenges in inductive synthesis. To combat this, the inductive synthesis we consider in this work allows increasing the memory locally and keeps the growth of the design space somewhat manageable. Note that increasing the memory might introduce symmetries in the design space as is highlighted in [4] and we can introduce symmetry reduction techniques to make the design spaces smaller, however, this topic will not be further discussed in this work as we will introduce our own memory injection strategy in the later chapters where the symmetry reduction did not prove to be helpful.

Let's first consider a memory injection strategy where we want to analyse families $\mathcal{F}_1^{\mathcal{M}}, \mathcal{F}_2^{\mathcal{M}}, ...\mathcal{F}_k^{\mathcal{M}}$ in this order. This means we first consider 1-state FSCs representing memoryless strategies, then we consider all 2-state FSCs and so on. This memory injection

strategy is very simple, however, from a practical point of view it's still useful as we cannot be sure what number of states we should start with and the analysis of small families like $\mathcal{F}_1^{\mathcal{M}}$ is cheaper and we will use the bounds learned in each stage to make the next ones faster. The size of the design space grows exponentially with respect to the number of memory nodes so exploring families with big k usually is not even feasible. Of course, it holds that $\mathcal{F}_1^{\mathcal{M}} \subseteq \mathcal{F}_2^{\mathcal{M}} \subseteq ... \subseteq \mathcal{F}_k^{\mathcal{M}}$ so by exploring bigger families we also explore the ones with smaller k. These two facts need to be considered when coming up with a memory injection strategy for inductive synthesis.

3.3 Simulation-based and Reinforcement Learning Methods

We give brief summaries for both simulation-based and reinforcement learning methods. They are used for planning in complex domains [26].

3.3.1 Simulation-based methods

Simulation-based methods can be used to solve POMDPs by simulating the environment and the agent's interactions with it. Specifically, the agent's policy is evaluated by running simulation runs of the POMDP and computing the expected reward obtained under that policy. One common simulation-based method for solving POMDPs is Monte Carlo Tree Search (MCTS) [27]. MCTS builds a search tree that represents possible sequences of actions and observations, and the expected rewards associated with each sequence. MCTS uses a simulation-based approach to estimate the value of each node in the tree and then selects actions to maximize the expected reward. Overall, simulation-based methods are useful for solving POMDPs because they allow the agent to evaluate policies without having to explicitly compute the probability distributions over states and observations. Instead, the agent can use simulations to estimate the expected reward associated with different actions and observations and then select the action that maximizes the expected reward.

3.3.2 Reinforcement Learning

Reinforcement learning (RL) can be used to solve POMDPs by iteratively interacting with the environment, receiving rewards based on its actions, and updating its policy based on the received rewards and observations. Approaches based on RL provide very strong scalability. One common RL algorithm for solving POMDPs is deep Q-learning [14]. In deep Q-learning, the agent maintains Q-values, which estimate the expected reward of taking a particular action in a given state. The Q-values are updated based on the observed rewards and transitions between states. Learning the Q-values for POMDPs this way is intractable because a Q-value would be needed for each possible belief or for arbitrary long observation-based histories. We can use a function approximator, such as a neural network, to approximate the Q-values. The Q-values can be parameterized by either the belief and the action or the observation-based history and the action. Sometimes RL uses MCTS (or other simulations) to estimate the expected rewards that can be used for the learning process. The overall goal of RL-based methods in POMDPs is to learn functions that can be used to choose actions for the current belief. For an overview of recent advances in the RL approach for POMDPs see [29].

Chapter 4

Limitations of State-of-the-art

In this chapter, we present the limitations of the state-of-the-art methods introduced in the previous chapter. Firstly, we look at point-based methods, simulation-based methods and reinforcement learning methods to explain why we do not consider them in the remainder of this work from the perspective of our problem statement. That is the perspective of offline synthesis in POMDPs with indefinite-horizon specifications. We also include a discussion on why FSCs are a go-to representation for POMDP schedulers. In the second part of this chapter, we present the limitations of belief-based methods with cut-off approximations and the inductive synthesis of FSCs on two small POMDPs. These examples show that even current state-of-the-art methods struggle with seemingly simple problems. This motivates the improvements we propose in the next chapter as we seek ways to improve both belief-based methods and inductive synthesis. We also introduce a synthetic POMDP called Lanes+ which showcases the struggles of both of the methods combined and serves as a motivation for closed-loop integration.

4.1 Limitation of Alternative Approaches

We focus on belief-based methods and inductive synthesis in this work. In this section, we give reasons as to why other methods are not suitable when we consider our problem statement. We also explain the importance of FSCs as a means of representing schedulers compared to other representations.

Point-based methods

Point-based methods allow us to efficiently approximate the value function. However, they have two main drawbacks. Firstly, they often require a discount factor to work and have a guarantee of fast convergence. We do not work with specifications that allow discounting in this work. Secondly, they represent the computed policy as a set of α-vectors. This set is not convenient to use. To compute what action to play, we need to remember our current belief and perform a number of vector multiplications to find out which α-vector produces the best value. Once we find the best α-vector for the current belief we can play the action assigned to this vector and perform a belief update. And for every new belief we have to repeat this process. You can compare this process to using FSCs. If we encode the functions of an FSC and store them in a hash table for example we can obtain what action to play in constant time for every observation-based history.

Simulation-based methods

Simulations allow us to obtain a lot of information about the model without the need to unfold the possibly complex belief space. This provides an advantage for the online planning problem. However, in this work, we focus on the offline synthesis problem. This means that a lot of simulations are required to obtain a good picture of what the policy should look like. This limits the effectiveness of simulations which are typically good for obtaining approximations quickly. Another problem is that simulations themselves do not provide a clear way how to obtain a full policy. That's why they are usually used in tandem with other methods.

Reinforcement learning

RL and other machine learning approaches proved useful for many practical POMDP problems. They tend to learn a neural network that is used to obtain what actions to choose, effectively it encodes the policy. They are able to solve big problems. Their drawback is that they are very intensive on computational resources and the computation itself is not easy to verify. The policy represented by a neural network is not easy to understand and verify. Compare that to the policies represented as FSCs, where each state clearly defines the considered actions and memory updates. This can be crucial when we want to use these policies in safety-critical systems. We believe that our focus on improving the formal approach to the synthesis problem will prove beneficial for such applications. Additionally, the produced neural networks can be pretty large.

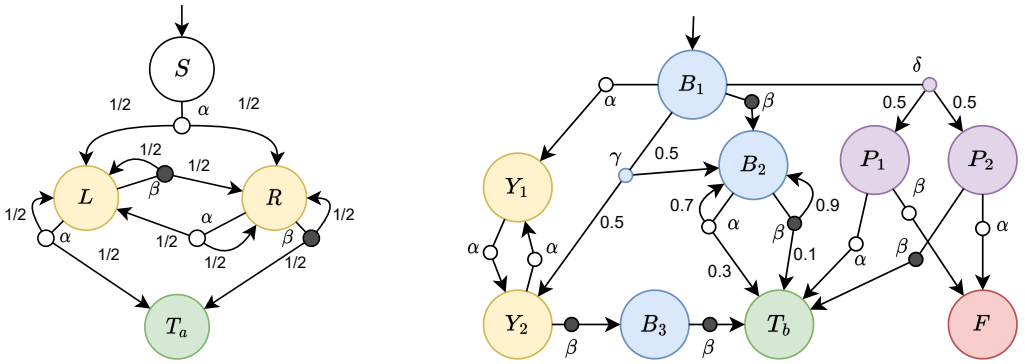
4.2 Limits of Belief-based Methods and Inductive Synthesis

We will first introduce a simple POMDP example where belief exploration struggles, then we give a small example where inductive synthesis struggles and we indicate how both of these approaches can help each other to overcome these struggles. By the combination of these simple POMDPs, we get a POMDP that is difficult to solve for either of the two approaches but a combined symbiotic approach can efficiently tackle this problem. We showcase the full potential of this symbiosis on a new synthetic POMDP called Lanes+.

Challenging POMDP for belief exploration approach

Consider POMDP from Figure 4.1a. Let's call it $\mathcal{M}_a$. The objective is to minimise the expected number of steps needed to reach target state T_a. There are two optimal policies for this specification in model $\mathcal{M}_a$. One policy always takes action α, the other policy always takes action β. Both policies yield 4 expected steps. An FSC realising one of these policies can be found using the policy search approach under 1s.

Let's analyse the structure of the belief MDP $\mathcal{M}_a^{\mathcal{B}}$ closer. The initial belief is $\{S \mapsto 1\}$. By taking action α (the case for taking action β is symmetric), 'yellow' observation is observed and the belief is updated to $\{L \mapsto \frac{1}{2}, R \mapsto \frac{1}{2}\}$. Detailed inspection shows that the set of reachable beliefs is infinite rendering $\mathcal{M}_a^{\mathcal{B}}$ to be infinite. The belief exploration can only construct a finite approximation of the belief space $\overline{\mathcal{M}_a^{\mathcal{B}}}$ by exploring $\mathcal{M}_a^{\mathcal{B}}$ up to some depth and applying cut-offs at the frontier states. We can then analyse this finite approximation using off-the-shelf tools yielding the minimising policy $\sigma_{\mathcal{B}}$ assigning to each belief state the optimal action.



(a) (b)

Figure 4.1: (a) and (b) contain two example POMDPs. Colours encode observations. Unlabelled transitions have probability 1. Omitted actions (e.g. γ, δ in state B_2) execute a self-loop. (a) contains 4 states, 3 observations and 5 actions. This model is used to showcase one of the main weaknesses of belief exploration. (b) contains 9 states, 5 observations and 20 actions. This model's structure causes issues for inductive policy search.

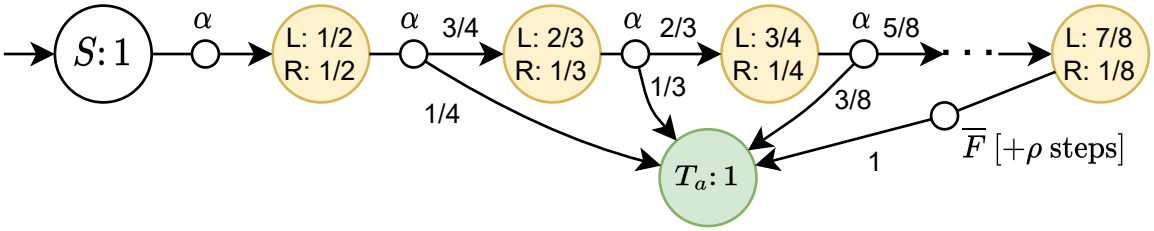


Figure 4.2: Markov chain (MC) induced by the minimising policy $\sigma_{\mathcal{B}}$ in the finite approximation $\overline{\mathcal{M}}_a^{\mathcal{B}}$ of the POMDP from Figure 4.1a. In the rightmost state, policy $\overline{F}$ is used to compute the cut-off value represented by ρ.

A simple way to compute cut-off values is to use an arbitrary controller $\overline{F}$ and compute the expected number of steps needed under $\overline{F}$. This operation is cheap if $\overline{F}$ is compact. Figure 4.2 shows a MC induced by $\sigma_{\mathcal{B}}$ in $\overline{\mathcal{M}}_a^{\mathcal{B}}$. The belief $\{L \mapsto \frac{7}{8}, R \mapsto \frac{1}{8}\}$ is cut off using controller $\overline{F}$. If we consider the belief exploration implemented in Storm [12], unfolding 1000 states and using heuristics to obtain controller $\overline{F}$, we obtain a sub-optimal controller $F_{\mathcal{B}}$ that reaches target in ≈ 4.1 steps. If we instead used better $\overline{F}$ (not necessarily optimal) computed by some efficient policy search implementation we would only need to explore a few states to beat $F_{\mathcal{B}}$. This might not seem like a big issue for the belief exploration, however, we can scale the model from Figure 4.1a to contain 100 intermediate (yellow) states and the belief space for this new model becomes much more complicated and the need for good approximation grows. If we use Storm to explore 1 million belief states for this new model and apply its heuristic controllers as cut-offs we obtain the value of over 2100 steps meanwhile the policy search is able to find an FSC with the optimal value of 98 steps almost instantly. Again we can use this FSC provided by policy search to help belief exploration construct a better finite approximation and also find the optimal strategy. Generally, structures similar to the one presented here can appear in practical models, and in some cases can render the belief exploration very inefficient.

Challenging POMDP for policy search approach

Consider POMDP $\mathcal{M}_b$ from Figure 4.1b. The objective is to minimise the expected number of steps to T_b. The reachable belief space for this model is finite, it consists of 9 states to be more precise, and therefore solving this POMDP is trivial for the methods using belief exploration. The optimal controller $\sigma_{\mathcal{B}}$ first picks action γ, on observing 'yellow' observation it plays β twice, otherwise it always picks α. An FSC with 3 memory nodes realises this strategy. The inductive policy search implemented in PAYNT [5] can find the optimal policy as well, however, it has to consult about 20 billion candidate policies. This requires 545 model-checking queries; the optimal FSC is found after 105 queries and the remaining queries prove that no better 3-state FSC exists.

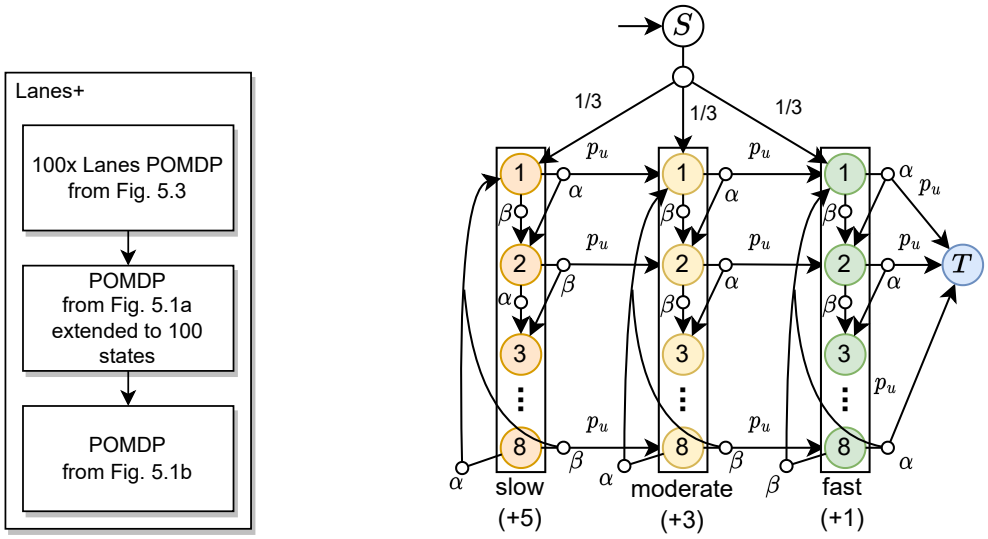
We can use reference policies to guide the policy search, the policy search we considered uses the fully observable MDP policy which in this case picks (senseless) action δ in B_1 first. This action is not good in $\mathcal{M}_b$, even though it's the optimal action in the underlying MDP, as we will not be able to avoid reaching fail state F from the 'purple' states as we cannot tell in which state we are exactly. Using policy $\sigma_{\mathcal{B}}$ obtained by the belief-based approach instead, action δ is not considered. As $\sigma_{\mathcal{B}}$ picks 3 different actions in states with 'blue' observation, we know that an FSC mimicking this strategy will need at least three memory nodes. This means we can skip a family of 2-FSCs. Using these facts in the inductive policy search we can reduce the total number of required model-checking queries by a factor of ten and we are able to find the optimal 3-state FSC after just 23 queries. This shows that considering better reference policies can lead to significant speedup. More importantly, it shows that considering (even non-optimal) POMDP policies has benefits compared to the fully observable policies which are standardly used.

4.2.1 The potential of symbiotic approach

Using the models from Figure 4.1 we can construct a model where both belief exploration and inductive policy search struggle, by putting the two models sequentially one after another. These combinations, however, are not enough to show that a closed-loop symbiotic approach can be beneficial. For this, we introduce one more synthetic model called Lanes+ which will be used in the experimental evaluation as well.

New POMDP Lanes+

The structure of the Lanes+ model is illustrated in Figure 4.3a. It is a sequential composition of a Lanes POMDP (see Figure 4.3b) repeated 100 times, followed by the POMDP from Figure 4.1a extended to 100 states, followed by the POMDP from Figure 4.1b. The core component – Lanes model – was designed with two main goals in mind: i) the optimal FSC $F_{\mathcal{I}}$ requires several memory nodes and ii) the model can be easily scaled up such that an exhaustive policy search is not feasible. When combining Lanes with POMDPs from Figure 4.1, we obtain a model which is difficult for both standalone approaches as well as their one-way integrations and the two-way symbiotic integration is required to find a good controller. The optimal FSC for the Lanes+ model requires 8 memory nodes. The corresponding family $\mathcal{F}_{\mu}^{\mathcal{M}}$ contains $\approx 10^{43}$ candidate controllers, which is unattainable for inductive search.



(a) (b)

Figure 4.3: (a) Overall structure of the Lanes+ POMDP. (b) The Lanes POMDP. The POMDP consists of three lanes and the objective is to cross the lanes from left to right with the minimum amount of actions. The lanes are cyclic meaning when the last state of a lane is reached the next state is the first state of that given lane. Playing actions in the slow lane gives you reward 5, moderate lane 3, and fast lane 1, so the faster you get to faster lanes the better. When performing the upgrading action (e.g. action α in the first state of the slow lane), the lane is upgraded with probability p_u; with probability $1-p_u$ the agent moves to the next state of the lane.

Symbiotic approach on Lanes+ model

We consider minimisation of the expected reward for the model Lanes+ with a 15-minute timeout. The belief-based approach implemented in Storm yields the value 18870, because the intermediate extended model from Figure 4.1a makes the reachable belief space difficult to explore. The policy search method implemented in PAYNT finds an FSC with 2 memory nodes achieving value 8223 and is unable to search the bigger families because of the state-space explosion. This sub-optimal FSC significantly improves the belief MDP approximation and enables the implementation from Storm to find a controller with value 6471. The symbiotic synthesis loop finds the optimal FSC with the value 4805 after two full iterations of the algorithm discussed in later chapters. This shows the potential of a symbiosis between inductive synthesis and belief exploration. In the experimental evaluation, we confirm this potential on more practical models.

Chapter 5

Integration of Inductive Synthesis and Belief-based Methods

This chapter starts by introducing two main novel ideas: i) we improve the approximations of the unexplored belief space, a crucial part of the belief-based approach, by using FSCs at cut-off beliefs ii) we improve the inductive synthesis approach by considering reference POMDP strategies obtained from other sources and using them to steer the search within the families of FSCs as well as creating more suitable families of FSCs. We then unify the theory of FSCs to include special cases of FSCs such as μ-FSCs of belief FSCs. We also give formulas on how to compute the sizes of FSCs for different types. These formulas will be used to compare the sizes of solutions obtained in the experimental section. Lastly, we present the main innovation proposed in this work. We introduce a novel symbiotic policy synthesis algorithm called Saynt. This algorithm iteratively combines the inductive search of FSCs and belief exploration. We presented a need for such combination in the previous chapter and we will present the improvements this algorithm brings in Chapter 6.

5.1 Using FSCs for Cut-off Values

Before we introduce the main idea of using FSCs for computing cut-off values, we showcase how we can derive FSCs from the belief MDPs. This process is important for our work as we focus on synthesising compact policies that are easy to use. We look at two cases considering finite and infinite belief MDPs.

Finite belief MDPs Let $T^{\mathcal{B}} := \{b \in \mathcal{B}_{\mathcal{M}} \mid O(b) = z^T\}$ denote the set of target beliefs. If the reachable state space is finite and we constructed the finite belief MDP $\mathcal{M}^{\mathcal{B}}$, we can use the standard model checking techniques to compute the memoryless policy $\sigma_{\mathcal{B}} : \mathcal{B}_{\mathcal{M}} \to Act$ that maximises $\mathbb{P}[b \models \Diamond T^{\mathcal{B}}]$ for each $b \in \mathcal{B}_{\mathcal{M}}$. We can translate this deterministic, memoryless policy $\sigma_{\mathcal{B}}$ into the corresponding FSC $F_{\mathcal{B}} = (\mathcal{B}_{\mathcal{M}}, b_0, \gamma, \delta)$ with action function $\gamma(b, z) = \sigma_{\mathcal{B}}(b)$ and update function $\delta(b, z, z') = \langle b \mid \sigma_{\mathcal{B}}(b), z' \rangle$ for all $z, z' \in Z$.

Infinite or not fully explored belief MDPs In case the reachable belief space is infinite or too large so we cannot unfold it completely, a finite approximation $\overline{\mathcal{M}^{\mathcal{B}}}$, based on techniques discussed in the previous chapters, is used instead. Let's have some set of explored beliefs $\mathcal{E}$ and the frontier beliefs $\mathcal{U}$. To complete the finite abstraction we will assign a cut-off value $\underline{V}(b)$ for each $b \in \mathcal{U}$. Ultimately, we define a finite MDP

$\overline{\mathcal{M}^{\mathcal{B}}} = (\mathcal{E} \cup \mathcal{U} \cup \{b_\top, b_\bot\}, b_0, Act, \overline{\mathbf{P}_{\mathcal{B}}})$ with the transition function: $\overline{\mathbf{P}_{\mathcal{B}}}(b, \alpha) := \mathbf{P}_{\mathcal{B}}(b, \alpha)$ for explored belief $b \in \mathcal{E}$ for all $\alpha \in Act$, and $\overline{\mathbf{P}_{\mathcal{B}}}(b, \alpha) := \{b_\top \mapsto \underline{V}(b), b_\bot \mapsto 1 - \underline{V}(b)\}$ for frontier beliefs $b \in \mathcal{U}$ and all $\alpha \in Act$, where $b_\top$ and $b_\bot$ are fresh sink states, i.e. they are absorbing for all actions $\alpha \in Act$. The reachable state space of $\overline{\mathcal{M}^{\mathcal{B}}}$ is finite and therefore a policy maximising $\mathbb{P}^{\overline{\mathcal{M}^{\mathcal{B}}}}_{\max}\left[\Diamond(T^{\mathcal{B}} \cup \{b_\top\})\right]$ induces an FSC for the original POMDP $\mathcal{M}$.

FSC Cut-offs

In Chapter 3, we discussed why and how to approximate the unexplored belief space. We also observed that finding good observation-based controllers is crucial to obtain good approximations. We can use an observation-based controller to compute suitable cut-off values. The closer the cut-off value is to the actual optimum in a belief, the better the approximation we obtain. In particular, if the cut-off values coincide with the optimal value, cutting off in the initial state is optimal. Obviously, finding a good (optimal) approximation this way is as hard as solving the original POMDP. We consider under-approximative value functions by applying any FSC to the POMDP and lifting the results to the belief MDP. So we want to find good a FSC to get good cut-off values. We note here that implementation in [7] considers simple memoryless FSCs. We generalise belief exploration with cut-offs such that arbitrary sets of FSCs are supported.

Let's consider an arbitrary, but fixed FSC $F_{\mathcal{I}} \in \mathcal{F}^{\mathcal{M}}$ for POMDP $\mathcal{M}$. Let $p_{s,n} := \mathbb{P}^{\mathcal{M}^{F_{\mathcal{I}}}}[(s, n) \models \Diamond T]$ for state $(s, n) \in S \times N$ in the corresponding induced MC. We denote the cut-off value $V(b, n) := \sum_{s \in S_{O(b)}} b(s) \cdot p_{s,n}$ for belief b and memory node n. This value corresponds to the probability of reaching a target state in $\mathcal{M}^{F_{\mathcal{I}}}$ when starting from state $s \in S$ according to the probability given by belief b with memory node $n \in N$. We define the overall cut-off value for belief b induced by controller F as $\underline{V}(b) := \max_{n \in N} V(b, n)$. We can clearly see that $\underline{V}(b) \leq \mathbb{P}^{\mathcal{M}^{\mathcal{B}}}_{\max}[b \models \Diamond T^{\mathcal{B}}]$. Computing $\underline{V}(b)$ for a given belief b is relatively simple as the values $p_{s,n}$ only need to be computed once. However, the complexity of the FSC-based cut-off approach depends on the size of the induced MC which means that it is essential that the FSC used to compute the cut-off values is concise.

Model checking the finite approximation MDP $\overline{\mathcal{M}^{\mathcal{B}}}$ with cut-off values induced by an FSC $F_{\mathcal{I}}$ yields a maximising memoryless policy $\sigma_{\mathcal{B}}$. We want to use the formalism of FSC to represent this policy. We will use $F_{\mathcal{B}}$ to denote this belief FSC. We construct $F_{\mathcal{B}}$ by considering both $F_{\mathcal{I}}$ used to compute cut-off values and the necessary memory nodes for each explored belief $b \in \mathcal{E}$. Concretely, we introduce a corresponding memory node for each explored belief. In each such node the action $\sigma_{\mathcal{B}}(b)$ is selected. For the memory update, we distinguish between two cases based on the next belief after executing action $\sigma_{\mathcal{B}}(b)$ in $\overline{\mathcal{M}^{\mathcal{B}}}$. Given $z' \in Z$, if the successor belief $b' = \langle b \mid \sigma_{\mathcal{B}}(b), z' \rangle \in \mathcal{E}$, the memory is updated to the corresponding node. Otherwise, $b' \in \mathcal{U}$ must hold, which means the successor is part of the frontier. The memory is then updated to the memory node n of FSC $F_{\mathcal{I}}$ that maximises the cut-off value $V(b', n)$. This corresponds to the notion of switching to the strategy represented by FSC $F_{\mathcal{I}}$ once a belief from the frontier is encountered. So overall we follow two policies, $\sigma_{\mathcal{B}}$ for explored belief and $F_{\mathcal{I}}$ from frontier beliefs onward. This is formalised as:

Definition 14 (Belief FSC with FSC cut-offs) *Let $F_{\mathcal{I}} = (N, n_0, \gamma_{\mathcal{I}}, \delta_{\mathcal{I}})$ and let $\overline{\mathcal{M}^{\mathcal{B}}}$ be the finite approximation defined as before. The belief-based FSC with cut-offs is $F_{\mathcal{B}} = (\mathcal{E} \cup N, b_0, \gamma, \delta)$ with action function $\gamma(b, z) = \sigma_{\mathcal{B}}(b)$ for $b \in \mathcal{E}$ and $\gamma(n, z) = \gamma_{\mathcal{I}}(n, z)$ for $n \in N$ and arbitrary $z \in Z$. The update function δ is defined for all $z, z' \in Z$ by*

$\delta(n, z, z') = \delta_{\mathcal{I}}(n, z, z')$ *if* $n \in N$, *and for* $b \in \mathcal{E}$ *with* $b' = \langle b \mid \sigma_{\mathcal{B}}(b), z' \rangle$ *by:*

$$\delta(b, z, z') = b' \text{ if } b' \in \mathcal{E}, \text{ and } \delta(b, z, z') = \text{argmax}_{n \in N} V(b', n) \text{ otherwise.}$$

This definition provides us with insight into how the actual strategy works, but if we want to compute the value of this strategy we want to construct induced MC. The reachable segment of the explored belief space with strategy $F_{\mathcal{B}}$ will represent the backbone of the induced MC. We have to evaluate the states from the frontier $\mathcal{U}$ and assign each state its cut-off value based on FSC $F_{\mathcal{I}}$. To do this we compute the probability $\mathbb{P}[s \models \Diamond T]$ (or rewards depending on our original specification) of reaching the target states when starting in state s and executing FSC $F_{\mathcal{I}}$ for each $s \in S$ in the original POMDP. Let's denote this value $p^{F_{\mathcal{I}}}(s)$. Then if we take a belief $b \in \mathcal{U}$ its value according to FSC $F_{\mathcal{I}}$ is $\sum_{s \in S} b(s) \cdot p^{F_{\mathcal{I}}}(s)$. If we add direct transitions with the computed probabilities from each frontier state to the target state we can directly get the value of the strategy by checking the MC given by the belief space exploration and controller $F_{\mathcal{B}}$ for specification $\mathbb{P}(\models \Diamond b_T)$.

5.2 Using Reference Policies to Improve Inductive Synthesis

We want to improve the inductive synthesis approach presented in Chapter 3. Consider the search for the optimal k-FSC $F \in \mathcal{F}_k^{\mathcal{M}}$ for POMDP $\mathcal{M}$. To accelerate the search for F within the given family, we can use a reference policy. One example of such policy is the policy $\sigma_{\mathcal{B}}$ extracted from a (finite approximation of the) belief MDP. This reference policy can be used to shrink the FSC family. For each observation $z \in Z$, we collect the set $Act[\sigma_{\mathcal{B}}](z) := \{\sigma_{\mathcal{B}}(b) \mid b \in \mathcal{B}_{\mathcal{M}}, O(b) = z\}$ of actions that were selected by $\sigma_{\mathcal{B}}$ in beliefs with observation z. Simply put, the set $Act[\sigma_{\mathcal{B}}](z)$ contains the actions used by the reference policy in states with observation z. It holds that $Act[\sigma_{\mathcal{B}}](z) \subseteq \text{Act}(z)$. These subsets are significant as they contain only the actions that are important for executing policy $\sigma_{\mathcal{B}}$. Since $\sigma_{\mathcal{M}}$ is an observation-based policy it provides valuable insight into what actions are likely not important for finding good policies. The better this reference policy is the better information we can extract, however, finding a good policy requires solving the POMDP which is a difficult problem as we already discussed in this work. Note here, that the policy $\sigma_{\mathcal{B}}$ can be replaced by any other observation-based policy. We want to focus on these important actions while searching for a good FSC by constructing a subset of FSCs $\{(N, n_0, \gamma, \delta) \in \mathcal{F}_k^{\mathcal{M}} \mid \forall n \in N, \forall z \in Z, \gamma(n, z) \in Act[\sigma_{\mathcal{B}}](z)\}$.

Restricting the action selection may exclude the optimal k-FSC and we need to keep this in mind. It also does not guarantee that the optimal FSC in the restricted family achieves the same value as the reference policy $\sigma_{\mathcal{B}}$ as replicating $\sigma_{\mathcal{B}}$ may require more memory nodes than we have currently available in the family, i.e. more than k memory nodes. We first search the restricted space of FSCs given by the reference policy before completely searching the rest of the design space. This also accelerates the search as the earlier a good policy is found, the easier it is to discard other candidate FSCs as we can prove that they are not optimal. Furthermore, in case the algorithm terminates earlier (notice the anytime aspect given in our problem statement), we are more likely to have found a reasonable policy.

Let's have two families of FSCs $\mathcal{F}_a$ and $\mathcal{F}_b$. Informally, we say that $\mathcal{F}_a$ is subfamily of $\mathcal{F}_b$ if every FSC in $\mathcal{F}_a$ is also in $\mathcal{F}_b$. Let's consider family $\mathcal{F}_k^{\mathcal{M}}$ and say $\mathcal{F}_i$ is one of its subfamilies. By $\overline{\mathcal{F}_i}$ we denote the complement subfamily, i.e. $\mathcal{F}_i \cap \overline{\mathcal{F}_i} = \emptyset$ and $\mathcal{F}_i \cup \overline{\mathcal{F}_i} = \mathcal{F}_k^{\mathcal{M}}$. To ensure we search a given family completely after we go through the main restricted family of FSCs $\mathcal{F}_m$, we have to also search subfamilies $\mathcal{F}_1, \mathcal{F}_2, ..., \mathcal{F}_j$ such that $\bigcup \mathcal{F}_i = \overline{\mathcal{F}_m}$. The number

of these subfamilies is equal to the number of restricted states in the MDP abstraction $MDP(\mathcal{F}_k^{\mathcal{M}})$. Let's say we have 2 restricted states s_0, s_1 in the abstraction MDP representing the same observation z with enabled actions $Act(s_0) = Act(s_1) = \{\alpha, \beta\}$. Assume that for our reference policy $\sigma_{\mathcal{B}}$ it holds that $Act[\sigma_{\mathcal{B}}](z) = \{\alpha\}$. Then our main restricted family only allows action α in both states. The first subfamily will allow action β in s_0 and both actions α, β in s_1. The lat subfamily will allow α in s_0 and β in s_1. This way we covered all possible FSCs from our initial family. This process is illustrated in Figure 5.1 on an example with three abstraction states.

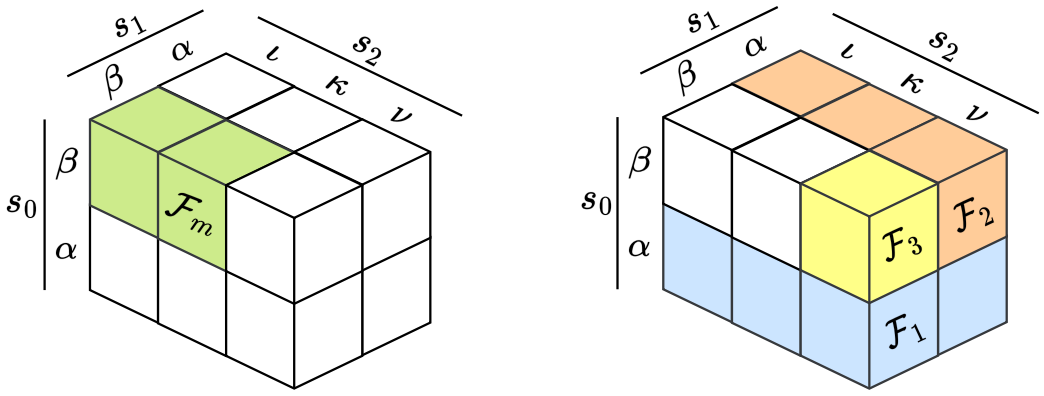


Figure 5.1: Let's say we have a reference policy $\sigma_{\mathcal{B}}$, which chooses action β for abstraction states s_0, s_1 and actions ι, κ for abstraction state s_1. Each cube represents one choice of actions for the abstraction states. (a) shows the main restricted family $\mathcal{F}_m$ given by policy $\sigma_{\mathcal{B}}$. Here we can see that the reference policy might significantly reduce the size of the important part of the design space. (b) shows the subfamilies that remain after we explored $\mathcal{F}_m$. We can clearly see that when we explore all the families $\mathcal{F}_m, \mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$ we explored the whole original family. We also see that we do not consider any action assignment twice. So it holds that $\overline{\mathcal{F}_m} = \mathcal{F}_1 \cup \mathcal{F}_2 \cup \mathcal{F}_3$ and $\mathcal{F}_i \cap \mathcal{F}_j = \emptyset$ for any two constructed subfamilies.

Improving Memory Injection Strategy

We can improve the memory injection strategy to obtain FSCs that are more compact and create families that are smaller but still contain the important FSCs. For this, let's define a new type of FSC:

Definition 15 (μ-FSC) *A memory model for POMDP $\mathcal{M}$ is a function $\mu : Z \to \mathbb{N}$. Let $k = max_{z \in Z} \mu(z)$. The k-FSC $F \in \mathcal{F}_k^{\mathcal{M}}$ with nodes $N = \{n_0, n_1, ..., n_{k-1}\}$ is a μ-FSC iff for all $z \in Z$ and for all $i > \mu(z)$ it holds: $\gamma(n_i, z) = \gamma(n_0, z)$ and $\delta(n_i, z, z') = \delta(n_0, z, z')$ for any $z' \in Z$.*

Let $\mathcal{F}_u^{\mathcal{M}}$ denote the family of all μ-FSCs for given memory function μ. Informally, we can say that the memory model μ dictates that for prior observation z only $\mu(z)$ memory nodes are utilised, while the rest of the nodes behave exactly as the default memory node n_0. If we are using a memory model μ where $\mu(z) < k$ for some observations $z \in Z$, we greatly reduce the number of candidate FSCs in the family. We will use μ-FSCs in our improved version of inductive synthesis as well as our symbiotic algorithm.

With μ-FSCs defined, we can consider an improved version of the memory injection strategy. The memory nodes are used to store information to improve decisions, however, if we have a state $s \in S$ with observation $z \in Z$ from POMDP $\mathcal{M}$ such that $\forall s' \in S : s = s' \iff O(s) = O(s')$ we know that when we see observation z on some path we are in state s and therefore we do not need to remember any information to make a decision for this observation. This allows us to reduce the design space. So the improved memory injection strategy starts with family $\mathcal{F}_1^{\mathcal{M}}$ but then creates a family $\mathcal{F}_u^{\mathcal{M}}$ where $\mu(z) = 1$ for all observations z where only one state in the model has this observation, otherwise $\mu(z) = 2$ and so on for bigger k.

We can improve this strategy further by using the reference policies to perform more educated decisions on where we need to add memory. To be more precise we can use sets $Act[\sigma_{\mathcal{B}}]$ to determine the k we want to search. If in some observation $z \in Z$ the belief policy $\sigma_{\mathcal{B}}$ uses $|Act[\sigma_{\mathcal{B}}](z)|$ distinct actions, then we for sure know that in order to enable the use of all of these actions, we require at least $k = \max_{z \in Z} |Act[\sigma_{\mathcal{B}}](z)$ memory nodes. This fact stems from the definition of FSC as the action mapping function γ can only choose one action for each (n, z) pair. However, we have to be careful with adding a lot of memory because it might become infeasible to search the constructed families even with our proposed restrictions. For this reason, we use a more refined view. We consider FSCs from family $\mathcal{F}_{\mu}^{\mathcal{M}}$ given by memory function μ as per definition 15. This helps us reduce the size of the abstraction even if the reference policy requires a lot of memory nodes.

Posterior-aware or posterior-unaware FSCs

The inductive synthesis presented in Chapter 3 considers posterior-unaware FSCs. As a reminder, posterior-unaware FSC is an FSC where $\delta(n, z, z') = \delta(n, z, z'')$ for all $n \in N$ and $z, z', z'' \in Z$. The advantage of using posterior-unaware FSCs is that the MDP abstraction corresponding to a family of posterior-unaware FSCs is smaller as fewer parameters have to be considered. It is easy to see that the posterior-unaware FSCs are a special case of the general posterior-aware ones. On the other hand, the posterior-aware FSCs typically need fewer memory nodes to denote the same strategy. This fact can sometimes be beneficial for the inductive synthesis approach.

5.3 FSC Overview

We wish to compare the sizes of all different controller types as one of our goals given in the problem statement is to find concise strategies. For FSC $F = (N, n_0, \gamma, \delta)$ we define its size $size(F) := size(\gamma) + size(\delta)$, i.e. the memory required to encode functions γ and δ[1]. The sizes for each FSC type are shown in Table 5.1. Before we showcase how to obtain the sizes of FSCs we want to discuss the flexibility of general FSCs compared to belief FSCs. Belief FSCs contain states that coincide with the explored beliefs $\mathcal{E}$. This means that in certain states only a limited amount of actions can be considered. If we change the POMDP such that the new reachable belief space does not correspond one-to-one with the states of belief FSC, this belief FSC becomes unusable as it would get stuck. Meanwhile, the more general FSCs produced by the inductive synthesis can be used as long as the set of observations and actions does not change in the POMDP. This also explains why we can use the more general form of a FSC to compute cut-off approximations.

[1]Remember: $\gamma : N \times Z \to Act$ and $\delta : N \times Z \times Z \to N$

FSC class	$size(\gamma)$	$size(\delta)$
k-FSC	$k \cdot \lvert Z \rvert$	$2 \cdot \sum_{n \in N} \sum_{z \in Z} \lvert post(n,z) \rvert$
μ-FSC	$\sum_{z \in Z} \mu(z)$	$2 \cdot \sum_{z \in Z} \sum_{i=0}^{\mu(z)-1} \lvert post(n_i,z) \rvert$
posterior-unaware μ-FSC	$\sum_{z \in Z} \mu(z)$	$\sum_{z \in Z} \mu(z)$
$F_{\mathcal{B}}$ using $F_{\mathcal{I}}$ for cut-offs	$size(\gamma_{\mathcal{I}}) + \lvert \mathcal{E} \rvert$	$size(\delta_{\mathcal{I}}) + 2 \cdot \sum_{b \in \mathcal{E}} \lvert post(b, O(b)) \rvert$

Table 5.1: Sizes for all different types of FSCs introduced in this work. The overall size of an FSC F is $size(F) = size(\gamma) + size(\delta)$.

Assume a POMDP $\mathcal{M}$, a k-FSC F with memory nodes $N = \{n_0, n_1, ..., n_{k-1}\}$ and an induced MC $\mathcal{M}^F = (S \times N, (s_0, n_0), P^F)$. Encoding the function γ of a general k-FSC requires $size(\gamma) = \sum_{n \in N} \sum_{z \in Z} 1 = k \cdot |Z|$ memory. Encoding function δ might require $k \cdot |Z|^2$, however, it's rare that in each state-memory pair (s, n) all posterior observations can be observed. We, therefore, encode $\delta(n, z, \cdot)$ as a sparse adjacency list. To define the size of such a list properly, consider the induced MC $\mathcal{M}^F$. Let $post(n, z) := \{O(s') \mid \exists s \in S_z : (s', \cdot) \in supp(\mathbf{P}^F(s, n))\}$ denote the set of posterior observations reachable when making a transition in a state (s, n) of $\mathcal{M}^F$ with $O(s) = z$. Then, $\delta(n, z, \cdot)$ can be encoded as a list $\{(z', \delta(n, z, z')) \mid z' \in post(n, z)\}$ of posterior observation and memory node pairs. Thus we get:

$$size(\delta) = \sum_{n \in N} \sum_{z \in Z} 2 \cdot |post(n, z)| = 2 \cdot \sum_{n \in N} \sum_{z \in Z} |post(n, z)|$$

If we use memory model μ instead of considering the full k-FSCs we have $\mu(z)$ distinct memory nodes for each observation $z \in Z$. Therefore the needed memory changes:

$$size(\gamma) = \sum_{z \in Z} \sum_{i=0}^{\mu(z)-1} 1 = \sum_{z \in Z} \mu(z)$$

$$size(\delta) = 2 \cdot \sum_{z \in Z} \sum_{i=0}^{\mu(z)-1} |post(n_i, z)|$$

If we consider posterior-unaware μ-FSC we can limit the size of the FSC further. For each $z \in Z$ and $n_i \in \{n_0, ..., n_{\mu(z)-1}\}$ we only need to store single value $\delta(n_i, z, \cdot)$. We get:

$$size(\delta) = \sum_{z \in Z} \sum_{i=0}^{\mu(z)-1} 1 = \sum_{z \in Z} \mu(z)$$

and $size(\gamma)$ remains the same as for the general μ-FSCs. Lastly, let's consider the belief FSC $F_{\mathcal{B}} = (\mathcal{E} \cup N, b_0, \gamma, \delta)$ obtained with applying FSC $F_{\mathcal{I}} = (N, n_0, \gamma_{\mathcal{I}}, \delta_{\mathcal{I}})$ at frontier states. Each non-frontier state $b \in \mathcal{E}$ is associated with the unique prior observation $O(b)$. Therefore, for every $b \in \mathcal{E}$ we must store exactly one action and a list $\{(z', \delta(b, O(b), z')) \mid z' \in post(b, O(b))\}$ of posterior observation and belief pairs. Of course, we have to also include the size of the FSC $F_{\mathcal{I}}$ used to compute the cut-off values at the frontier states. Overall, we obtain:

$$size(\gamma) = size(\gamma_{\mathcal{I}}) + \sum_{b \in \mathcal{E}} 1 = size(\gamma_{\mathcal{I}}) + |\mathcal{E}|$$

$$size(\delta) = size(\delta_{\mathcal{I}}) + 2 \cdot \sum_{b \in \mathcal{E}} |post(b, O(b))|$$

5.4 Symbiotic Policy Synthesis

We introduce a symbiotic approach for finding good FSCs for POMDPs with indefinite-horizon specifications. Our symbiotic approach combines tools PAYNT [5] and STORM [12] which represent two state-of-the-art tools for this problem. The important underlying notions of these tools were introduced in Chapter 3. We would like to note here that the potential advantages this symbiosis brings are not strictly tied to these two tools and might be applicable to other combinations of approaches. We call our symbiotic approach SAYNT in the remainder of this work. Firstly, we introduce the overall structure of SAYNT showcasing how these tools can work together and then we provide the anytime symbiotic algorithm for finding good FSCs.

5.4.1 Overall framework

The main idea behind the symbiotic approach is essentially the fact that a policy found by one of the methods can be used to boost the other method either in the sense of finding a policy with better value or finding said policy faster. The key observation is that these policies can be beneficial even if they are sub-optimal in terms of the objective at hand. This is crucial as both methods are trying to tackle the same difficult problem and therefore both of them might fail to find near-optimal solutions on their own and symbiosis might be the key. We've shown this potential in Section 4.2.1. The symbiotic approach combining PAYNT and STORM is sketched in Figure 5.2. The orange part represents the inner workings of the inductive synthesis implemented in PAYNT. PAYNT works with families of FSCs. It constructs MDP abstraction for a given family and performs MDP model checking on the abstraction. If the specification is violated we know that the whole family violates it and we can prune it. If the specification holds we obtain some potentially good policy but we have to check if this policy is observation-based or not. If this policy is not observation-based we have to refine the current family to obtain better abstraction. If the policy is observation-based we've found new improving FSC $F_{\mathcal{I}}$ and we can update the specification. This part is described in detail in Section 3.2. The blue part represents belief exploration from STORM. STORM tries to find POMDP policies by unfolding belief space. It unfolds a finite fragment of the belief MDP and computes heuristic bounds that are used to approximate unexplored belief space. This process produces a finite approximation of the complete belief MDP. We can perform MDP model checking on this approximation and the obtained policy is a valid observation-based FSC $F_{\mathcal{B}}$. A detailed explanation is provided in Section 3.1. The red arrows highlight where and how the two methods communicate. How the methods use the provided policies to boost their respective performance is explained in the previous sections of this chapter. In short, the FSCs $F_{\mathcal{I}}$ obtained by policy search are used to guide the partial belief MDP to the target, allowing us to compute values of unexplored belief states. Meanwhile, the FSCs $F_{\mathcal{B}}$ obtained by the belief exploration are used to shrink the set of considered policies and steer the abstraction as we are able to extract better information compared to the fully-observable MDP strategies.

5.4.2 Anytime Symbiotic Algorithm

We now use the structure shown in Figure 5.2 as a base for our algorithm. In the previous sections, we showed how inductive synthesis can benefit from policies obtained by belief exploration and how belief exploration can use FSCs produced by inductive synthesis to achieve better results. A natural is to use improved inductive synthesis results to further

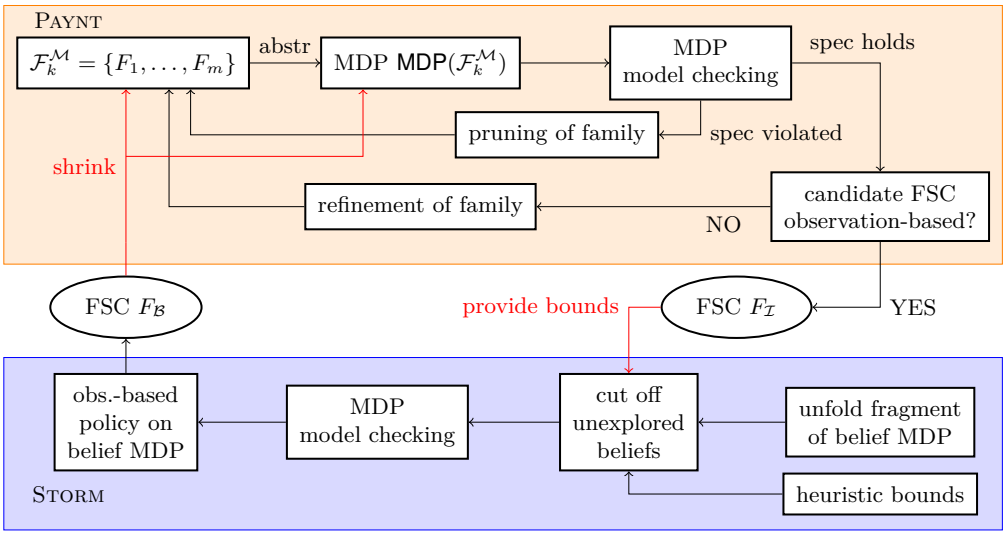


Figure 5.2: Schematic depiction of the symbiotic approach. The orange part represents the inductive synthesis implemented in PAYNT and the blue part represents belief exploration implemented in STORM. The red arrows show how the two approaches communicate using their FSCs.

improve belief exploration and improved belief exploration to provide even better reference policies to the inductive synthesis, i.e. to alternate between these approaches and in each step let them share the found FSCs. This symbiotic approach is captured in Algorithm 1.

We iterate until a global timeout t is reached. In each iteration, we make both controllers available to the user as soon as they are computed. We start the symbiotic algorithm with the inductive mode (l. 3-8), where we initially consider 1-FSCs represented in $\mathcal{F}_\mu^{\mathcal{M}}$. Method `search`(l. 8) investigates given family $\mathcal{F}$ and outputs the new maximising FSC $F_{\mathcal{I}}$ (if it exists). If the timeout $t_{\mathcal{I}}$ interrupts the synthesis process, the method additionally outputs yet unexplored parameter assignments $\mathcal{F}$. If family $\mathcal{F}$ is fully explored within the timeout $t_{\mathcal{I}}$ (l. 4), we increase k and continue the synthesis with a new family of FSCs. After the timeout $t_{\mathcal{I}}$, we run the belief exploration method `explore`for $t_{\mathcal{B}}$ seconds, where we use FSC $F_{\mathcal{I}}$ for computing cut-off values. $F_{\mathcal{I}}$ is used in tandem with the heuristic strategies computed by belief exploration. After the timeout $t_{\mathcal{B}}$ is reached the exploration stores its current configuration (so that it knows where to continue in the next iteration). Then the exploration proceeds to obtain the cut-off values at unexplored states using $F_{\mathcal{I}}$ and computes the optimal policy $\sigma^{\mathcal{M}^{\mathcal{B}}}$ from which we extract the belief FSC $F_{\mathcal{B}}$ incorporating the FSC $F_{\mathcal{I}}$. Before we continue with the next iteration we check whether the belief FSC gives better value than $F_{\mathcal{I}}$ and whether this FSC gives any reason to update the memory model (l. 10). If we determine that the memory should be updated, i.e. whether there's any observation for which the current memory value is smaller than the number of distinct actions considered by $\sigma_{\mathcal{B}}$, we update μ and continue with new family $\mathcal{F}$ (l. 11-12).

Now we would like to highlight some of the decisions we had to make in the design of this algorithm. We wanted a simple push-button algorithm that is powerful for the general class of POMDPs and the choices we made led to this goal as we will show in the next chapter. These decisions were made based on our initial experiments with the integration. The main decisions we discuss here: i) the order of the used methods in the algorithm, ii) the timeout for inductive search and belief exploration, iii) complete search vs. family pruning,

Algorithm 1: Anytime symbiotic algorithm for finding FSCs for POMDPs

Input : POMDP $\mathcal{M}$, set T of target states, timeout values $t, t_{\mathcal{I}}, t_{\mathcal{B}}$
Output : Best FSCs $F_{\mathcal{I}}$ and $F_{\mathcal{B}}$ found so far

$F_{\mathcal{I}} \leftarrow \bot$, $\mathcal{F} \leftarrow \mathcal{F}_1^{\mathcal{M}}$, $k \leftarrow 0$, $\mu \leftarrow \{z \mapsto 1 \mid z \in Z\}$, $F_{\mathcal{B}} \leftarrow \bot$, $\sigma_{\mathcal{B}} \leftarrow \bot$
while *not timeout* t **do**
  while *not timeout* $t_{\mathcal{I}}$ **do**
    if $\mathcal{F} = \emptyset$ **then**
      $k \leftarrow k + 1$
      $\forall z \in Z \colon \mu(z) \leftarrow \max\{\mu(z), k\}$
      $\mathcal{F} \leftarrow \mathcal{F}_{\mu}^{\mathcal{M}}$
    $\mathcal{F}, F_{\mathcal{I}} \leftarrow$ `search`$(\mathcal{F}, F_{\mathcal{I}}, Act[\sigma_{\mathcal{B}}]$ **if** $\mathbb{P}^{\mathcal{M}^{F_{\mathcal{I}}}}[\Diamond T] > \mathbb{P}^{\mathcal{M}^{F_{\mathcal{B}}}}[\Diamond T]$ **else** $\bot)$
  $\sigma_{\mathcal{B}}, F_{\mathcal{B}} \leftarrow$ `explore`$(t_{\mathcal{B}}, F_{\mathcal{I}})$
  if $\mathbb{P}^{\mathcal{M}^{F_{\mathcal{I}}}}[\Diamond T] \leq \mathbb{P}^{\mathcal{M}^{F_{\mathcal{B}}}}[\Diamond T]$ *and* $\exists z \in Z \colon \mu(z) < |Act[\sigma_{\mathcal{B}}](z)|$ **then**
    $\forall z \in Z \colon \mu(z) \leftarrow |Act[\sigma_{\mathcal{B}}](z)|$
    $\mathcal{F} \leftarrow \mathcal{F}_{\mu}^{\mathcal{M}}$
  yield $F_{\mathcal{I}}, F_{\mathcal{B}}$

iv) considering non-FSC cut-offs in reference policies. The implementation of our algorithm allows the user to set the pruning of non-main subfamilies and consider the non-FSC cut-offs (discussed below) if they want but we will not consider these settings in the remainder of this work. The timeouts we used for our experiments will be discussed in the experimental evaluation chapter.

The order of the methods

Starting the symbiotic iteration with policy search is the preferred way to combine these methods. While it is possible to start the loop with belief exploration, it does not provide any lasting advantages and might in fact hinder the performance. The reason is that if the initial belief-based method produces a bad policy then it might negatively impact the inductive synthesis by steering the search to unimportant families that contain bad FSCs. Finding bad $F_{\mathcal{I}}$ then further slows down the improvement of the belief-based method as the cut-off values are not good. On the other hand, bad FSCs obtained by inductive synthesis do not hinder the belief exploration in any way and actually still provide better bounds than the heuristically computed strategies in most cases.

Setting the timeouts

An important part of the algorithm is the setting of the two timeouts $t_{\mathcal{I}}$ and $t_{\mathcal{B}}$. We leave these values as parameters in the algorithm as it's easy to see that these have a great impact on the overall result. But there are general rules on how to set these. The inductive synthesis usually takes longer to obtain good FSCs on more complex models. Also, the belief exploration timeout does not include the time needed to obtain the belief policy, i.e. the belief MDP model checking. These facts mean that we usually want to have $t_{\mathcal{I}} > t_{\mathcal{B}}$. We also want to ensure that multiple iterations are allowed to finish as we have shown in our motivation section and will show in experiments. Therefore the sum of these timeouts should be at least 3-4x less than the overall time we want to run the algorithm for.

Complete search of the families

We discussed the use of reference policies to help us find good FSCs more quickly by prioritizing the search in the main restricted subfamily. One approach would be to only focus on these main families in the inductive synthesis, search in them quickly and go to bigger families quickly. However, we outlined that it might be important to perform the complete search, i.e. also searching for solutions in the other subfamilies as showcased in Figure 5.1. The reason for this is that the reference policy might not be entirely correct and can have some biases that come from the method used to obtain them. The complete search ensures that even when the reference policy is bad we still get a good FSC and might outperform the method used to get the reference policy.

Considering non-FSC cut-offs

The belief-based method from Storm computes memoryless probabilistic policies to be used at cut-offs which we replace by FSCs computed by inductive synthesis. We've observed that the small FSCs computed very quickly by inductive synthesis are able to beat these policies but sometimes one of these policies might be good for some belief and we are left with the choice of including this policy in the context of the reference policy. The difficult part is that they are probabilistic and it's hard to quantify what actions are actually important. That's why we do not consider these policies by default even if they produce good cut-off values for some of the frontier beliefs. This in most cases leads to better results as the main family is more true to the reference policy and contains only the important restrictions.

Chapter 6

Experimental Evaluation

In this chapter, we present an experimental evaluation of the proposed improvements from the previous chapter. We first introduce the considered POMDPs and discuss the setup of the experiments. We then showcase our proposed ideas for improving cut-off approximations in belief-based methods and improving inductive synthesis work with just simple additional computations. These results directly motivate the proposed symbiotic algorithm which is evaluated at the end of this chapter. We focus on our main goal, i.e. finding good FSCs quickly. However, we also evaluated the memory usage of the algorithm, the sizes of the found FSCs and we also discuss the advantage of producing two FSCs in SAYNT. With the results presented in this chapter, we show that our proposed improvements and especially the symbiotic algorithm SAYNT beats the current state-of-the-art methods for the synthesis problem in POMDPs with indefinite-horizon specifications.

6.1 Selected Benchmarks and Experiments Setup

With the focus being on indefinite-horizon specifications, our baselines are the recent belief exploration technique [7] implemented in STORM [17] and the inductive synthesis method [4] implemented in PAYNT [5]. PAYNT uses STORM for parsing the input POMDPs and for model checking MDPs, but not for solving POMDPs. Our symbiotic framework (see Figure 5.2 and Algorithm 1) has been implemented on top of PAYNT and STORM, combining them in a closed-loop fashion. In the rest of this chapter, we will use STORM and PAYNT to refer to the implementation of belief exploration and inductive synthesis respectively. We will use SAYNT to refer to our proposed symbiotic framework.

Setup All of the experiments are run on a single core of a machine equipped with an Intel i5-12600KF @4.9GHz CPU and 64GB of RAM. PAYNT searches for posterior-unaware FSCs using abstraction refinement (explained in Section 3.2), as suggested by [4]. The default setting of STORM is to apply cut-offs as presented in Section 3.1. SAYNT uses the default settings for both PAYNT and STORM. The parameters from Algorithm 1 were set to $t_{\mathcal{I}} = 60s$ and $t_{\mathcal{B}} = 10s$ by default. At the end of this chapter, we discuss how changing these values affects the performance.

Benchmarks We consider models from AI and formal verification communities obtained from [4, 6, 7, 9]. We evaluate the methods on a selection of these models, supplemented by larger variants of these models (Drone-8-2 and Refuel-20), by one model from [15] (Milos-97)

Model	$\|S\|$	$\sum Act$	$\|Z\|$	Spec.	Over-approx.	Model	$\|S\|$	$\sum Act$	$\|Z\|$	Spec.	Over-approx.
4x3-95	22	82	9	$R_{\max}$	≤ 2.24	Drone-4-2	1226	2954	761	$P_{\max}$	≤ 0.98
4x5x2-95	79	310	7	$R_{\max}$	≤ 3.26	Drone-8-2	13k	32k	3195	$P_{\max}$	≤ 0.99
Hallway	61	301	23	$R_{\min}$	≥ 11.5	Lanes+	2741	5285	11	$R_{\min}$	≥ 4805
Milos-97	165	980	11	$R_{\max}$	≤ 80	Netw-3-8-20	17k	30k	2205	$R_{\min}$	≥ 4.31
Network	19	70	5	$R_{\max}$	≤ 359	Refuel-06	208	565	50	$P_{\max}$	≤ 0.78
Query-s3	108	320	6	$R_{\max}$	≤ 600	Refuel-20	6834	25k	174	$P_{\max}$	≤ 0.99
Tiger-95	14	50	7	$R_{\max}$	≤ 159	Rocks-12	6553	32k	1645	$R_{\min}$	≥ 17.8

Table 6.1: Information about the benchmark POMDPs.

and by the synthetic model explained in Section 4.2.1 (Lanes+). We excluded benchmarks for which PAYNT or STORM finds trivial (expected) optimal solutions in a matter of seconds. The reason for this is that if PAYNT or STORM finds a solution in a trivial fashion, SAYNT is also able to find and the comparison is pointless. The benchmarks were selected to illustrate the advantages as well as the drawbacks of all three synthesis approaches: belief exploration, inductive search, and the symbiotic technique. Table 6.1 lists for each POMDP the number of states $|S|$, the total number of actions $\sum Act := \sum_s |Act(s)|$, the number of observations $|Z|$, the specification (either maximising or minimising a reachability probability P or expected reward R), and a known over-approximation on the optimal value computed using the technique from [6]. These over-approximations are solely used as rough estimates of the optimal values and for most of these models, we do not know how close to the optimum we are.

6.2 Evaluation of the One-way Integrations

We want to showcase that the main ideas presented in Chapter 5 can be used to improve the state-of-the-art tools. We first look at improving belief approximations by using FSC cut-offs. Secondly, we showcase the improvements in the inductive synthesis with belief-based reference policies.

6.2.1 FSCs Improving Approximations of the Belief MDP

In these experiments. PAYNT is used to obtain a sub-optimal FSC $F_{\mathcal{I}}$ within 10s (or the first found if a bigger time limit is needed) which is then used by STORM to compute better cut-offs. We ask the question (Q1): *Do the FSCs from inductive synthesis raise the accuracy of the belief MDP?* Table 6.2 (left) lists the results. From these results, we get the following observation:

Observation 1 *Belief exploration can yield better FSCs (and sometimes even faster) using FSCs from PAYNT even if the latter FSCs are from optimal.*

For instance, STORM with provided $F_{\mathcal{I}}$ finds an FSC with value 0.97 for the Drone-4-2 benchmark within a total of 10s (1s+9s for obtaining $F_{\mathcal{I}}$), compared to obtaining an FSC of value 0.95 in 56s on its own. A value improvement is also obtained if STORM runs longer. For the Network model, the value improves with 37% (short-term) and 47% (long-term) respectively, at the expense of investing 3s to find $F_{\mathcal{I}}$. For the other models, the relative improvement ranges from 3% to 25%. A further value improvement can be achieved when

Model	PAYNT $F_{\mathcal{I}}$	Short STORM	Short STORM + $F_{\mathcal{I}}$	Long STORM	Long STORM + $F_{\mathcal{I}}$
Drone-4-2	0.94	0.92	0.97	0.95	0.97
$P_{\max}$	9s	1s	1s	56s	57s
Network	266.1	186.7	274.5	202.1	277.1
$R_{\max}$	3s	<1s	<1s	26s	33s
Drone-8-2	0.9	0.6	0.96	0.68	0.97
$P_{\max}$	28s	3s	3s	101s	103s
4x3-95	1.66	1.62	1.82	1.84	1.88
$R_{\max}$	7s	<1s	<1s	60s	72s
Query-s3	425.2	417.4	430.0	419.6	432.0
$R_{\max}$	7s	2s	2s	91s	94s
Milos-97	31.56	37.15	39.15	38.35	40.64
$R_{\max}$	3s	<1s	<1s	42s	42s
Hallway	16.05	13.07	12.63	12.55	12.55
$R_{\min}$	9s	1s	1s	160s	167s
Rocks-12	42	38	31.89	20*	20*
$R_{\min}$	<1s	<1s	<1s	10s	10s

Model	STORM $F_{\mathcal{B}}$	PAYNT	PAYNT + $F_{\mathcal{B}}$
4x5x2-95	2.08	0.94	2.03
$R_{\max}$	<1s	258s	38s
Refuel-20	0.09	<0.01	0.19
$P_{\max}$	1s	10s	11s
Tiger-95	50.38	2.99	28.73
$R_{\max}$	<1s	14s	23s
4x3-95	1.62	1.75	1.84
$R_{\max}$	<1s	14s	238s
Refuel-06	0.67	0.35	0.67
$P_{\max}$	<1s	<1s	42s
Milos-97	37.15	31.56	39.29
$R_{\max}$	<1s	3s	215s
Netw-3-8-20	11.93	11.07	10.95
$R_{\min}$	1s	185s	271s
Rocks-12	38	42	38
$R_{\min}$	<1s	<1s	<1s

Table 6.2: **Left (Q1)**: Experimental results on how a (quite sub-optimal) FSC $F_{\mathcal{I}}$ computed by PAYNT within 10s impacts STORM. (For Drone-8-2, the largest model in our benchmark, we use 30s). The "PAYNT" column indicates the value of $F_{\mathcal{I}}$ and its run time. The "Short STORM„ column runs storm for 1s and compares the value of FSC $F_{\mathcal{B}}$ found by STORM alone to STORM using $F_{\mathcal{I}}$. The "Long STORM" column is analogous, but with a 300s timeout for STORM. In the last row, * indicates that clipping was used. **Right (Q2)**: Experimental results on how an FSC $F_{\mathcal{B}}$ obtained by a shallow exploration of the belief MDP impacts the inductive synthesis by PAYNT. The "STORM„ column reports the value of $F_{\mathcal{B}}$ computed within 1s. The "PAYNT" column compares the values of the FSCs $F_{\mathcal{I}}$ obtained by PAYNT itself to PAYNT using the FSCs $F_{\mathcal{B}}$ within a 300s timeout.

using better FSCs $F_{\mathcal{I}}$ from PAYNT; see Section 6.3. Sometimes, belief exploration does not profit from $F_{\mathcal{I}}$. For Hallway, the unexplored part of the belief MDP becomes insignificant rather quickly, and so does the impact of $F_{\mathcal{I}}$. Clipping [7], a computationally expensive extension of cut-offs, is beneficial only for Rocks-12, rendering $F_{\mathcal{I}}$ useless. Though even in this case, using $F_{\mathcal{I}}$ significantly improves Short STORM that did not have enough time to apply clipping.

6.2.2 Belief-based FSCs Improving Inductive Synthesis

In these experiments, we run STORM for at most 1s, and use the result as a reference policy in PAYNT. We ask the question (Q2): *Does the exploration of the belief MDP boost the inductive synthesis of FSCs?* Table 6.2 (right) lists the results. From these results we make another important observation:

Observation 2 *Inductive synthesis can find much better FSCs (and sometimes much faster) when using FSCs obtained from belief exploration as reference policies to steer the search.*

For instance, for the 4x5x2 benchmark, an FSC is obtained about six times faster while improving the value by 116%. On some larger models, PAYNT alone struggles to find any good $F_{\mathcal{I}}$ and using $F_{\mathcal{B}}$ boosts this; e.g., the value for the Refuel-20 model is raised by a

factor 20 at almost no run time penalty. For the Tiger benchmark, a value improvement of 860% is achieved (albeit not as good as $F_{\mathcal{B}}$ itself) at the expense of doubling the run time. Thus: *even a shallow exploration of the belief MDP pays off in the inductive synthesis.* The inductive search typically profits even more when exploring the belief MDP further. This is demonstrated, e.g., in the Rocks-12 model: using the FSC $F_{\mathcal{B}}$ computed using clipping (see Table 6.2 (left)) enables PAYNT to find FSC $F_{\mathcal{I}}$ with the same (optimal) value 20 as $F_{\mathcal{B}}$ within 1s. Similarly, for the Milos-97 model, running STORM for 45s (producing a more precise $F_{\mathcal{B}}$) enables PAYNT to find an FSC $F_{\mathcal{I}}$ achieving a better value than controllers found by STORM or PAYNT alone within the timeout. (These results are not reported in the tables.) However, as opposed to Q1, where a better FSC $F_{\mathcal{I}}$ naturally improves the belief MDP, longer exploration of the belief MDP does not always yield a better $F_{\mathcal{I}}$: a larger $\overline{\mathcal{M}^{\mathcal{B}}}$ with a better $F_{\mathcal{B}}$ may yield a larger memory model μ, thus inducing a significantly larger family where PAYNT struggles to identify good FSCs.

6.3 Evaluation of the Anytime Symbiotic Algorithm

We know focus on evaluating the proposed symbiotic approach. We ask the question (Q3): *Is the symbiotic approach improving run time, controller's values and size?* Thus the goals of the experiments presented in this section are to investigate whether the symbiotic approach improves the run time (can FSCs of a certain value be obtained faster?), the memory footprint (how is the total memory consumption of the methods affected?), the controller's value (can better FSCs be obtained with the same computational resources?), and the controller's size (are more compact FSCs obtained?).

Value of the synthesized FSCs

Figure 6.1 plots the value of the FSCs produced by STORM, PAYNT, and SAYNT versus the computation time. Note that for maximal objectives, the aim is to obtain a high value (the first 4 plots) whereas for minimal objectives a lower value prevails. From the plots, we make the following observation:

Observation 3 *The FSCs from the symbiotic approach are superior in value to the ones obtained by the standalone approaches.*

The relative improvement of the value of the resulting FSCs differs across individual models, similar to the trends in Q1 and Q2. When comparing the best FSC found by STORM or PAYNT alone with the best FSC found by SAYNT, the improvement ranges from negligible (4x3-95) to around 3%-7% (Netw-3-8-20, Milos-97, Query-s3) and sometimes goes over 40% (Refuel-20, Lines+). We note that the distance to the (unknown) optimal values remains unclear. Recall that SAYNT returns two FSCs: $F_{\mathcal{I}}$ found during the inductive phase and $F_{\mathcal{B}}$ (that incorporates $F_{\mathcal{I}}$) found during the belief phase. That's why there are two lines associated with the results of SAYNT. The FSC value never decreases but sometimes does also not increase, as indicated by Hallway and Rocks-12 (see also Section 6.2.2). Some plots in Fig. 6.1 also include the FSC value by the one-shot combination of STORM and PAYNT. By comparing these FSCs we observe:

Observation 4 *SAYNT can improve the FSC value over the one-shot combinations whose ideas are presented in Sections 5.1 and 5.2.*

This is illustrated in, e.g., the 4x3-95 and Lanes+ benchmarks, see the 1st and 3rd plots in Figure 6.1 (left).

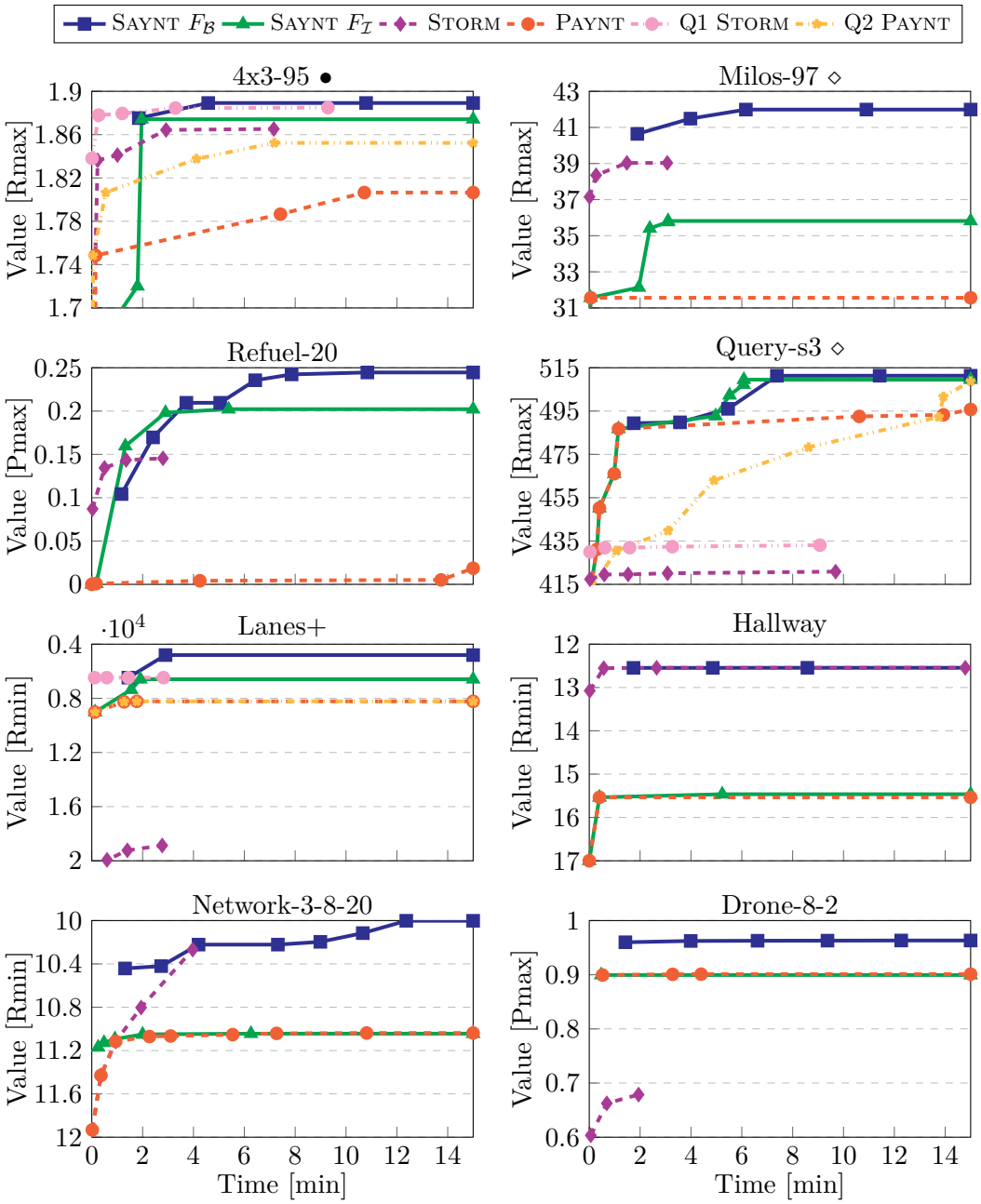


Figure 6.1: Value of the generated FSCs over time. We compare SAYNT with our two baselines STORM and PAYNT. Note that SAYNT produces two FSCs (blue and green lines). We also include comparisons with the proposed one-way improvements discussed in the previous sections 6.2.1 (Q1) and 6.2.2 (Q2). The lines ending before the timeout indicate that the 64GB memory limit was hit. • indicates that PAYNT and SAYNT synthesized posterior-aware FSCs. ◇ indicates that SAYNT ran with $t_{\mathcal{I}}$ =90s.

Total synthesis time

From Figure 6.1 we see that SAYNT initially needs some time to complete the first iteration (one inductive and one belief phase) in Algorithm 1 and thus during the beginning of the synthesis process, the standalone tools may provide FSCs of a certain value faster. However, we observe that:

Observation 5 *After the first iteration, SAYNT typically provides better FSCs in a shorter time and quickly overtakes both baselines.*

For instance, for the Refuel-20 benchmark SAYNT swiftly overtakes STORM after the first iteration. The only exception is Rocks-12 (discussed in the previous sections), where SAYNT with the default settings needs significantly more time than STORM to obtain an FSC of the same value.

Memory footprint

Belief exploration typically has a large memory footprint: STORM quickly hits the 64GB memory limit on exploring the belief MDP. This is indicated in the various plots of Fig. 6.1 by the purple dashed line that ends before the timeout. However, if we consider SAYNT we observe:

Observation 6 *SAYNT reduces the memory footprint of STORM alone by a factor of 3 to 4 while producing better FSCs.*

See Figure 6.2 for the comparison of average memory usage between STORM and SAYNT. The average memory footprint of running PAYNT standalone quickly stabilises around 700MB-1GB (for the largest models). The memory footprint of SAYNT is thus dominated by the restricted exploration of the belief MDP.

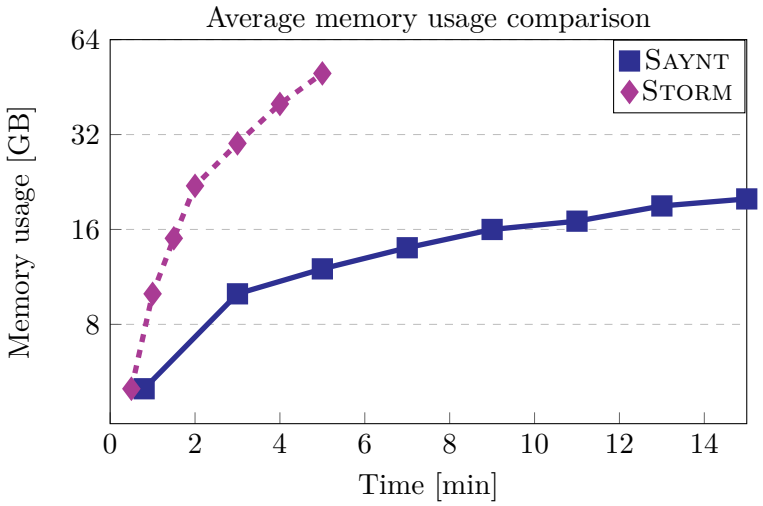


Figure 6.2: Comparison of average memory usage between STORM and SAYNT. The y-axis is logarithmic. We observe that SAYNT dramatically reduces the memory footprint of the belief exploration despite the fact that it produces better FSCs.

Models:	Lanes+	Hallway	Netw-3-8-20	Query-s3◇	Refuel-06	Drone-8-2	Refuel-20
$F_{\mathcal{B}}$	4805/8.1k	12.55/2k	10/40k	511.32/7.7k	0.67/84	0.96/237k	0.24/1.5k
$F_{\mathcal{I}}$	6591/34	15.46/86	11.04/4.8k	509.49/26	0.67/156	0.90/6.4k	0.2/362

Table 6.3: Trade-offs between the value and size in the resulting FSCs $F_{\mathcal{I}}$ and $F_{\mathcal{B}}$ found by SAYNT. Each cell reports value/size. The first three models have a minimising objective. ◇ indicates that SAYNT ran with $t_{\mathcal{I}}$ =90s.

The size of the synthesised FSCs

For selected models, Tab. 6.3 shows the trade-offs between the value and size of the resulting FSCs $F_{\mathcal{I}}$ and $F_{\mathcal{B}}$ found by SAYNT. From the experiments we observe:

Observation 7 *The FSCs $F_{\mathcal{I}}$ provided by the inductive synthesis in SAYNT are typically about one to two orders of magnitude smaller than the belief-based FSCs $F_{\mathcal{B}}$ with only a small penalty in their values*

There are models (e.g. Refuel-06) where a very small $F_{\mathcal{B}}$, having an even slightly smaller size than $F_{\mathcal{I}}$, does exist. The integration mostly reduces the size of $F_{\mathcal{B}}$ due to the better approximation of the belief MDP by up to a factor of two. This reduction has a negligible effect on the size of $F_{\mathcal{I}}$. This observation further strengthens the usefulness of SAYNT that jointly improves the value of $F_{\mathcal{I}}$ and $F_{\mathcal{B}}$. Hence, SAYNT gives users a unique opportunity to run a single, time-efficient synthesis and select the FSC according to the trade-off between its value and size.

6.3.1 More Results

In the previous sections, we presented results that helped us answer all of the important questions surrounding our proposed ideas and the proposed algorithm SAYNT. We showcased that the ideas from sections 5.1 and 5.2 can be used to obtain strong one-way integrations that outperform the base methods. We then showed that the proposed symbiotic algorithm SAYNT outperforms our baselines from multiple perspectives. Table 6.4 shows more results from our experiments, where we can compare the sizes of FSCs from different methods and the impact of using default versus non-default settings.

6.3.2 Customising the SAYNT Setup

In contrast to the stand-alone approaches as well as to the one-way integrations presented in Section 6.2, we observe that:

Observation 8 *SAYNT provides a single synthesis method that is efficient for a general class of models without tuning its parameters.*

Although SAYNT includes parameters that affect the interplay between the standalone approaches (see Algorithm 1 with the paragraphs discussing design choices and the paragraph on parameter settings), the experiments confirm that the default strategy provides superior and stable performance across all considered models. Below, we discuss some results when using a non-default setup, see the captions in Figure 6.1, Table 6.3, and Table 6.4. Using posterior-aware FSCs mostly significantly slows down the synthesis process. For Network and 4x3-95, it does improve the value of the default posterior-unaware FSCs by 2% to 4%.

For the former model, a better $F_{\mathcal{I}}$ also improves $F_{\mathcal{B}}$ about a similar value. In some cases (e.g. Query-s3), it is beneficial to increase the parameter $t_{\mathcal{I}}$ to 90s, giving PAYNT enough time to search for a good FSC $F_{\mathcal{I}}$ (the relative improvement is 6%) which also improves the value of the resulting FSC $F_{\mathcal{B}}$ about a similar value. $t_{\mathcal{I}}$ and $t_{\mathcal{B}}$ also determine if the size or the value of the FSCs is preferred. A detailed analysis of the experimental results suggests that it is more effective to invest time into searching for $F_{\mathcal{I}}$ approximating the belief MDP rather than into its deeper exploration.

Benchmark		Model Size		PAYNT		STORM		SAYNT			
Model	Spec.	S/Act	Z	$F_{\mathcal{I}}$	Size	$F_{\mathcal{B}}$	Size	$F_{\mathcal{B}}$	Size	$F_{\mathcal{I}}$	Size
4x3 95	$R_{\max}$	22 82	9	1.81 764s	36	1.87 414s	999	**1.89** 283s	968	1.87• 120s	126
4x3 95	$R_{\max}$	22 82	9	1.81 764s	36	1.87 414s	999	**1.89** 303s	869	1.79 678s	36
4x5x2 95	$R_{\max}$	79 310	7	0.94 305s	26	2.08 3s	102	2.08 71s	102	2.03 378s	38
Drone 4-2	$P_{\max}$	1226 3026	761	0.95 900s	1.5k	0.95 110s	135k	**0.97** 194s	140k	0.94 1s	1.5k
Drone 8-2	$P_{\max}$	13k 32k	3195	0.9 260s	6.4k	0.68 98s	280k	**0.96** 247s	140k	0.9 30s	6.4k
Hallway	$R_{\min}$	61 301	23	15.54 26s	66	12.55 916s	1.9k	12.55 263s	1.8k	15.46 293s	86
Lanes+	$R_{\min}$	2741 5289	11	8223 118s	42	18870 376s	8.1k	**4805** 173s	8.1k	6591 114s	34
Milos-97	$R_{\max}$	165 980	11	31.56 4s	40	39.03 88s	823	**41.99**◇ 370s	692	35.82◇ 185s	40
Milos-97	$R_{\max}$	165 980	11	31.56 4s	40	39.03 88s	823	**41.55** 270s	290	35.41 114s	40
Network	$R_{\max}$	19 70	5	280.33 38s	22	209.71 110s	2.4k	**289.18**• 395s	2k	287.23• 106s	54
Network	$R_{\max}$	19 70	5	280.33 38s	22	209.71 110s	2.4k	**284.51** 85s	1.8k	280.33 41s	22
Netw 3-8-20	$R_{\min}$	17k 30k	2205	11.04 638s	4.4k	10.27 238s	64k	**10** 742s	38k	11.04 379s	4.8k
Query s3	$R_{\max}$	108 320	6	502.3 931s	28	420.11 184s	12.9k	**511.32**◇ 566s	7.7k	509.49◇ 362s	26
Query s3	$R_{\max}$	108 320	6	502.3 931s	28	420.11 184s	12.9k	482.21 700s	7.7k	478.59 610s	28
Refuel 06	$P_{\max}$	208 565	50	0.35 <1s	100	0.67 182s	343	0.67 178s	84	0.67 84s	156
Refuel 20	$P_{\max}$	6834 24k	174	0.02 922s	348	0.15 468s	1.2k	**0.24** 386s	1.5k	0.2 173s	360
Tiger 95	$R_{\max}$	14 50	7	7.93 547s	34	50.38 <1s	58	50.38 71s	58	31.61 513s	48

Table 6.4: The quality and size of resulting FSCs provided by PAYNT, STORM, and SAYNT within the 15-minute timeout. The run times indicate the time needed to find the best FSC. Non-default settings: • marks experiments where PAYNT synthesized posterior-aware FSCs, ◇ marks experiments where integration parameter $t_{\mathcal{I}}$ was set to 90 seconds.

Chapter 7

Final Considerations

7.1 Future Research

This work opens up a lot of avenues for further research in the POMDP synthesis. We proposed two ideas for improving belief approximations and the inductive policy search respectively. In the case of belief approximations, we showcased that using simple FSCs can greatly improve state-of-the-art approaches. It is desirable to explore other methods and see if there are possible ways to improve them using these simple FSCs as they are compact and hold a lot of information about the POMDP and its specification. In the case of improved inductive synthesis, we can look into extracting more information from the reference policies. Currently, we only look at the considered set of actions for different observations but there might be more information about for example the needed memory, or the dependencies between different actions in the reference policies. We also might consider reference policies from different sources. As we said the idea of using a reference policy is not tool specific, however, in the experimental evaluation we only considered reference policies from belief exploration in Storm.

If there's any progress in these mentioned areas, we directly improve the proposed symbiotic algorithm Saynt. While this algorithm already outperforms the state-of-the-art we believe that further improvements are possible with more fine-tuned parameters. It might be also interesting to try and come up with different integrations e.g. including simulation-based and reinforcement learning methods. The motivation here is the fact that the POMDP synthesis question is a difficult one, and while coming up with new ideas on how to tackle it is important, there will always be some inherent flaws that might only be avoided by looking at the problem from multiple angles at once.

In this work, we considered indefinite-horizon specifications. The AI community prefers using infinite-horizon specifications with discounting. In many cases, the algorithms only work if a discount factor strictly less than 1 is used. We believe this topic surrounding the use of a discount factor is not explored properly. We think there are real-world practical problems where the use of a discount factor is inappropriate. While we can set the discount very close to 1, we observed in our initial experiments (not included in this work) that by doing so the performance of the algorithms drops to a point where the benefits of approaches that support indefinite-horizon specifications become clear. Future research in this area should focus on explaining the need for a discount factor and how the methods considering different specifications can help each other.

This work tries to bridge the safety-scalability gap that currently occurs in the POMDP synthesis problem. We believe that the discussion on this topic is an important one.

Reinforcement learning and other machine learning approaches provide good scalability, however, offer little safety guarantees compared to formal methods. This problem is prominent in a lot of areas of modern computer science and we believe that future research in the POMDP synthesis problem should focus on this important problem.

Finally, SAYNT produces two FSCs $F_{\mathcal{I}}$ and $F_{\mathcal{B}}$ and we observed interesting trade-offs between the size and the quality of these FSCs (see Observation 7), where their values can be close but their sizes vary a lot. This indicates that sometimes the produced FSCs are unnecessarily large. Future research might focus on developing algorithms for the minimisation of the FSCs. This could lead to an approach where we find a good FSC first and then make it as compact as possible using some external algorithm.

7.2 Conclusions

This work focused on the synthesis problem in POMDPs with indefinite-horizon specifications. We considered two main state-of-the-art approaches for this problem: i) belief exploration with cut-off approximations implemented in tool STORM, ii) the inductive synthesis of FSCs implemented in tool PAYNT. We showed clear limitations of state-of-the-art approaches on simple POMDP examples. We used the shown limitations to directly motivate two novel ideas. Firstly, we can improve the belief exploration by using FSCs to obtain better cut-off approximations. Secondly, we can improve the inductive synthesis approach by using information from reference policies to steer the search. With these ideas, it became clear that an interplay between belief exploration and inductive synthesis is possible. We proposed SAYNT, a symbiotic closed-loop integration of the two main approaches for controller synthesis in POMDPs. Using a wide class of models, we demonstrated that SAYNT substantially improves the value of the resulting controllers. It does so while also reducing the memory footprint of the belief exploration and improving the overall synthesis time. SAYNT works in an iterative fashion. After each iteration, two FSCs $F_{\mathcal{I}}$ and $F_{\mathcal{B}}$ are produced. This gives the user a unique choice between the small FSC $F_{\mathcal{I}}$ or better but much larger FSC $F_{\mathcal{B}}$. Another advantage of SAYNT is that it provides a single synthesis method for a general class of models without the need for tuning its parameters. This work strengthens the position of formal methods for the POMDP synthesis problem.

This thesis served as a basis for an article called "Search and Explore: Symbiotic Policy Synthesis in POMDPs". This article was accepted to International Conference on Computer Aided Verification 2023 (CAV'23), a CORE A* conference, in May 2023.

Bibliography

[1] Amato, C., Bernstein, D. S. and Zilberstein, S. Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs. *Autonomous Agents and Multi-Agent Systems*. Springer. 2010, vol. 21, no. 3, p. 293–320.

[2] Amato, C., Bonet, B. and Zilberstein, S. Finite-State Controllers Based on Mealy Machines for Centralized and Decentralized POMDPs. In: *AAAI*. AAAI Press, 2010, p. 1052–1058.

[3] Andriushchenko, R., Češka, M., Junges, S. and Katoen, J.-P. Inductive synthesis for probabilistic programs reaches new horizons. In: Springer. *TACAS*. 2021, vol. 12651, p. 191–209. LNCS.

[4] Andriushchenko, R., Češka, M., Junges, S. and Katoen, J.-P. Inductive Synthesis of Finite-State Controllers for POMDPs. In: *UAI*. PMRL, 2022, vol. 180, p. 85–95.

[5] Andriushchenko, R., Češka, M., Junges, S., Katoen, J.-P. and Stupinskỳ, Š. PAYNT: a tool for inductive synthesis of probabilistic programs. In: Springer. *CAV*. 2021, vol. 12759, p. 856–869. LNCS.

[6] Bork, A., Junges, S., Katoen, J.-P. and Quatmann, T. Verification of indefinite-horizon POMDPs. In: Springer. *ATVA*. 2020, vol. 12302, p. 288–304. LNCS.

[7] Bork, A., Katoen, J.-P. and Quatmann, T. Under-Approximating Expected Total Rewards in POMDPs. In: *TACAS (2)*. Springer, 2022, vol. 13244, p. 22–40. LNCS.

[8] Carr, S., Jansen, N., Wimmer, R., Serban, A., Becker, B. et al. Counterexample-Guided Strategy Improvement for POMDPs Using Recurrent Neural Networks. In: *International Joint Conference on Artificial Intelligence*. August 2019, p. 5532–5539. DOI: 10.24963/ijcai.2019/768.

[9] Cassandra, T. *Tony Cassandra's POMDP File Repository Page* [https://pomdp.org/examples/]. Accessed: 2023-01-23.

[10] Češka, M., Jansen, N., Junges, S. and Katoen, J.-P. Shepherding hordes of Markov chains. In: Springer. *TACAS*. 2019, vol. 11428, p. 172–190. LNCS.

[11] Cubuktepe, M., Jansen, N., Junges, S., Marandi, A., Suilen, M. et al. Robust Finite-State Controllers for Uncertain POMDPs. In: *AAAI*. AAAI Press, 2021, p. 11792–11800.

[12] Dehnert, C., Junges, S., Katoen, J.-P. and Volk, M. A Storm is Coming: A Modern Probabilistic Model Checker. In: *CAV*. Springer, 2017, vol. 10427, p. 592–600. LNCS.

[13] Hansen, E. A. Solving POMDPs by searching in policy space. In: *UAI*. 1998, p. 211–219.

[14] Hausknecht, M. and Stone, P. Deep Recurrent Q-learning for Partially Observable MDPs. In: *2015 AAAI Fall Symposium Series*. 2015.

[15] Hauskrecht, M. Value-Function Approximations for Partially Observable Markov Decision Processes. *J. Artif. Int. Res.* AI Access Foundation. aug 2000, vol. 13, no. 1, p. 33–94. ISSN 1076-9757.

[16] Heck, L., Spel, J., Junges, S., Moerman, J. and Katoen, J. Gradient-Descent for Randomized Controllers Under Partial Observability. In: *VMCAI*. Springer, 2022, vol. 13182, p. 127–150. LNCS.

[17] Hensel, C., Junges, S., Katoen, J.-P., Quatmann, T. and Volk, M. The probabilistic model checker Storm. *International Journal on Software Tools for Technology Transfer*. august 2022, vol. 24, p. 1–22. DOI: 10.1007/s10009-021-00633-z.

[18] Horak, K., Bosansky, B. and Chatterjee, K. Goal-HSVI: Heuristic Search Value Iteration for Goal POMDPs. In: *IJCAI*. AAAI Press, 2018, p. 4764–4770.

[19] Junges, S., Jansen, N., Wimmer, R., Quatmann, T., Winterer, L. et al. Finite-state Controllers of POMDPs via Parameter Synthesis. In: *UAI*. 2018, p. 519–529.

[20] Junges, S., Katoen, J., Pérez, G. A. and Winkler, T. The complexity of reachability in parametric Markov decision processes. *J. Comput. Syst. Sci.* 2021, vol. 119, p. 183–210.

[21] Kurniawati, H., Hsu, D. and Lee, W. S. SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces. In: *Robotics: Science and Systems*. MIT Press, 2008.

[22] Kwiatkowska, M. Z., Norman, G. and Parker, D. PRISM 4.0: Verification of Probabilistic Real-Time Systems. In: *CAV*. Springer, 2011, vol. 6806, p. 585–591. LNCS.

[23] Madani, O., Hanks, S. and Condon, A. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artificial Intelligence*. 2003, vol. 147, no. 1, p. 5–34.

[24] Mykel J. Kochenderfer, K. H. W. *Algorithms for Decision Making*. MIT Press, 2022. ISBN 9780262047012.

[25] Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 1994.

[26] Schrittwieser, J. et al. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*. Springer Science and Business Media LLC. dec 2020, vol. 588, no. 7839, p. 604–609. DOI: 10.1038/s41586-020-03051-4.

[27] Silver, D. and Veness, J. Monte-Carlo planning in large POMDPs. In: *NIPS*. Curran Associates, Inc., 2010, p. 2164–2172.

[28] Smallwood, R. D. and Sondik, E. J. The optimal control of partially observable Markov processes over a finite horizon. *Oper. Res.* INFORMS. 1973, vol. 21, no. 5, p. 1071–1088.

[29] Xiang, X. and Foo, S. Recent Advances in Deep Reinforcement Learning Applications for Solving Partially Observable Markov Decision Processes (POMDP) Problems: Part 1—Fundamentals and Applications in Games, Robotics and Natural Language Processing. *Machine Learning and Knowledge Extraction*. 2021, vol. 3, no. 3, p. 554–581. DOI: 10.3390/make3030029. ISSN 2504-4990.