An Idiomatic Framework for Cognitive Robotics

Damgaard, Malte Rørmose

DOI (link to publication from Publisher):
10.54337/aau520799670

Publication date:
2022

Document Version
Publisher's PDF, also known as Version of record

Link to publication from Aalborg University

Citation for published version (APA):
Damgaard, M. R. (2022). *An Idiomatic Framework for Cognitive Robotics*. Aalborg Universitetsforlag. Ph.d.-serien for Det Tekniske Fakultet for IT og Design, Aalborg Universitet https://doi.org/10.54337/aau520799670

AN IDIOMATIC FRAMEWORK FOR COGNITIVE ROBOTICS

BY
MALTE RØRMOSE DAMGAARD

DISSERTATION SUBMITTED 2022

AALBORG UNIVERSITY
DENMARK

An Idiomatic Framework for Cognitive Robotics

Ph.d. Dissertation

Malte Rørmose Damgaard

Dissertation submitted November, 2022

Dissertation submitted:	November 2022
PhD supervisor:	Prof. Thomas Bak Aalborg University
Assistant PhD supervisor:	Asst. Prof. Rasmus Pedersen Aalborg University
PhD committee:	Associate Professor Simon Bøgh (chairman) Aalborg University, Denmark
	Professor Richard J. Duro Fernández Universidade da Coruña, Spain
	Professor Volker Krueger Lund University, Sweden
PhD Series:	Technical Faculty of IT and Design, Aalborg University
Department:	Department of Electronic Systems

ISSN (online): 2446-1628
ISBN (online): 978-87-7573-781-9

Published by:
Aalborg University Press
Kroghstræde 3
DK – 9220 Aalborg Ø
Phone: +45 99407140
aauf@forlag.aau.dk
forlag.aau.dk

Printed in Denmark by Stibo Complete, 2023

Abstract

"I alone cannot change the world, but I can cast a stone across the water to create many ripples."
- Mother Teresa

By increasingly taking over dull, dirty, and dangerous jobs, robots have demonstrated that they have the potential to transfigure the world we live in. However, the ability of robots to effectively deal with unpredictable and dynamic environments by learning and reasoning from experience is still relatively limited. This constitutes a major barrier to the development of robots that can integrate fully and seamlessly into human societies. The objective of this ph.d. study has been to aid roboticists in this development by designing a framework providing unifying standards for implementing artificial cognition for robots with characteristics of human cognition that can easily be shared and reused.

The main contribution of this ph.d. study is a proposal of a design for such a framework. The topic of this dissertation is the process undertaken to reach the proposed design, which has been structured through the design cycle. The properties of the framework are investigated in a series of papers describing different empirical studies considering both simulation-based and real-world single-case mechanism experiments. These empirical studies cover concepts related to the framework, such as real-time computation, distributed computations, multi-robot systems, low-level reactive attention mechanisms, and appraisal-driven decision mechanisms.

The results of these empirical studies are generalized into a design theory stating that the framework under given assumptions can provide implementations of artificial cognition for robots related to decision-making. The design theory further states several advantageous properties of using the framework for such implementations.

In the end, it is concluded that with this study important steps have been taken towards a framework providing unifying standards for implementing artificial cognition for robots.

Resumé

"Jeg kan ikke ændre verden alene, men jeg kan kaste en sten gennem vandet for at skabe mange krusninger."

- Moder Teresa

Ved i stigende grad at overtage kedelige, beskidte og farlige arbejdsopgaver har robotter demonstreret, at de har potentialet til at forvandle verdenen, som vi kender den. Desværre er robotters evne til effektivt at håndtere uforudsigelige og dynamiske miljøer ved at lære af og tage beslutninger, på baggrund af erfaring, stadig relativt begrænset. Dette udgør en kæmpe barriere for udviklingen af robotter, der gnidningsløst kan integreres ind i menneskelige samfund. Formålet med dette ph.d.-projekt har været at gøre denne opgave lettere ved at designe et sæt af retningslinjer, der foreskriver standarder, som kan forene implementeringer af kunstig kognition til robotter med karakteristika, der minder om menneskelig kognition samt gøre det nemmere at dele og genbruge sådanne implementeringer.

Hovedbidraget fra dette ph.d.-projekt er et designforslag til sådanne et sæt af retningslinjer. Emnet der adresseres i denne afhandling, er den designproces, der blev udført for at nå frem til det foreslåede sæt af retningslinjer. Egenskaberne for det foreslåede sæt af retningslinjer er undersøgt igennem en række videnskabelige artikler, der beskriver forskellige empiriske studier omhandlende single-case mekanisme eksperimenter i simulering og i den virkelige verden. Disse empiriske studier behandler fx emnerne realtidsberegning, distribuerede beregninger, systemer med mere end en robot, reaktive opmærksomhedsmekanismer og beslutningsmekanismer drevet af robottens interne vurdering af dens omstændigheder.

Resultaterne af disse empiriske studier generaliseres til en designteori, der siger, at retningslinjerne under givne antagelser kan resultere i implantationer af kunstig kognition til robotter relateret til beslutningstagen. Derudover påpeger designteorien flere fordelagtige egenskaber, der kan opnås ved at bruge retningslinjerne til sådanne implementeringer.

Til slut konkluderes det, at der med dette projekt er taget vigtige skridt imod et sæt af retningslinjer, der foreskriver standarder, som kan forene implementeringer af kunstig kognition til robotter.

Contents

Preface

This dissertation is submitted as a collection of papers in partial fulfillment of the requirements for the degree of ph.d. at the *Department of Electronic Systems, Section of Automation and Control, Aalborg University, Denmark*. The study covered by this dissertation was carried out in the period from June 2019 to November 2022 as part of the strategic initiative of the Technical Faculty of IT and Design at Aalborg University called "The Human-Robot Interaction project".

The dissertation is structured in two parts. The first part serves as an introduction to the study and the scientific methodology used within this study since this usually cannot easily be conveyed within the scope of scientific papers. This part is divided into four chapters. The first chapter gives the motivation for the study together with an overview of the research objective and methodology. The second chapter summarizes the background information and the State-of-the-Art review used for problem investigation and requirement specification. The third chapter summarizes how each of the papers published or submitted as part of this study has contributed to the study and the design theory that has been formed from them. The fourth chapter concludes the dissertation. The second part of the dissertation consists of the four papers published or submitted as part of this study.

I would like to express my deepest gratitude for the proficient supervision carried out by my two supervisors Thomas Bak and Rasmus Pedersen. Throughout my study and research, you have provided indispensable discussions and opinions keeping me on the right track despite my, at times, excessive ambitions. Furthermore, I am grateful for the financial support from "The Human-Robot Interaction project" giving me the freedom and opportunity to pursue research at and beyond the bleeding edge of robotics.

Malte Rørmose Damgaard
Aalborg University, November 30, 2022

Part I

Summary

Chapter 1

Introduction

1.1 Motivation

For a long time, it has been prophesied that robots will have an immense impact on human societies by taking over dull, dirty, and dangerous jobs. They have been imagined to aid in tasks such as search-and-rescue, last-mile delivery, rehabilitation, housekeeping, eldercare, surveillance, and autonomous space exploration. The list of possible application areas seems endless, and already back in the 90s leading roboticists at the time concluded that

"time is ripe for the development of AI-based commercial service robots that assist people in everyday life" [1].

Unfortunately, except for robot vacuum cleaners and lawnmowers, robots have only truly had a breakthrough in the industry. What might be the reason for this mismatch between what was thought to be possible 40 years ago, and what has been shown to be possible?

When compared to traditional industrial environments, the biggest reason for the missing breakthrough of robotic technology in environments such as commercial and home spaces is probably that these spaces are notoriously unstructured and dynamic environments[1]. This is to a large extent due to the required interaction with humans, since humans not only are *"dynamic and non-deterministic, but also perceive(s) the robot, adapt(s) their own plans and actions"* [2]. A fact that was also pointed out in [3]. Combining this with the statement

"it is impossible to pre-program robots with every needed skill or even to predict all use cases. As a result, robots need the ability to learn new skills after they are deployed" [2],

[1]Why Indoor Robots for Commercial Spaces Are the Next Big Thing in Robotics

gives a hint to one possible reason why robots have not yet conquered these environments and fulfilled the 40-year-old prophecy. Today's state-of-the-art algorithms in robotics can indeed be combined to construct functional robots performing specific structured tasks in static environments. However, compositions of such highly specialized algorithms usually require hand-tuning, rigid interfaces between algorithms, and prior knowledge about all the possible situations to which the robot might be exposed. Consequently, such robotic systems will inherently lack the ability to evolve with and adapt to their environment, making them unsuitable for highly dynamic environments perfused by uncertainty due to required interaction with humans. This is in contrast to human's cognitive ability to generalize prior knowledge to overcome impasses and thereby robustly adapt to changing environments and new problems. On the other hand, the growing amount of robot applications that have shown to be successful during the last decade can presumably be attributed exactly to the possibility of easily sharing and reusing state-of-the-art algorithms across the entire robotic community as nurtured by idiomatic standardizations such as the meta-operating systems ROS and ROS2. This yields quite a conundrum: how is it possible to reap the benefits of idiomatic standardization while being able to develop robot cognition that can evolve with and adapt to unstructured, dynamic, and uncertain environments?

1.2 Research Objective and Methodology

Motivated by the considerations set forward in Section 1.1, this project takes its outset on the hypothesis that

Hypothesis:

A framework providing unifying standards for implementing artificial robot cognition with characteristics of human cognition that can easily be shared and reused will finally allow roboticists to develop robots that can conquer unstructured and dynamic environments perfused by uncertainty from human interaction.

This hypothesis cannot be verified without such a framework for implementing parts of artificial robot cognition, and therefore it naturally gives rise to the following technical research problem.

Technical Research Problem:

How to design a framework providing unifying standards for implementing artificial robot cognition with characteristics of human cognition that can easily be shared and reused so that roboticists can develop robots for unstructured and dynamic environments perfused by uncertainty from human interaction?

Examination of all aspects of the main hypothesis or the related technical research problem is outside the scope of a single ph.d. study. However, this should not stop a curious and aspiring researcher from pursuing answers. Therefore, the research objective of this ph.d. study has been to get as good an answer as possible to the technical research problem within the limited period of the study. As a result, this study has been carried out as curiosity-driven exploratory research conducted to get a better understanding of the technical research problem.

According to [4] the goal of any technical research problem is to design a treatment for solving some problem so that stakeholders can achieve their goal in a specific problem context. Such treatment not only consists of a designed artifact but also the desired interaction between that artifact and the problem context. It is not the designed artifact itself that solves the problem, but rather the interaction between the designed artifact and a problem context. In the above technical research problem, the artifact that needs to be designed is the "framework", the stakeholders are "roboticists", the goal of the stakeholders is to "develop robots", and the problem context is "unstructured and dynamic environments perfused by uncertainty from human interaction". The above technical research problem also specifies some requirements that the artifact should satisfy, namely that it should provide "unifying standards for implementing artificial robot cognition with characteristics of human cognition that can easily be shared and reused". According to [4] the design of such treatment is usually carried out as iterations of the design cycle comprised of the following task:

- problem investigation,
- treatment design,
- and treatment validation.

Usually, the problem context, the stakeholders, and the goals of the stakeholders are not clear at an early stage of a project. Therefore, the purpose of the problem investigation is to get a better understanding of all elements of the problem to be treated. Based on the problem investigation, it should be possible to derive a set of requirements based on which a treatment can be designed. When the treatment has been designed, it can be validated in models of the problem context. Based on the validation of the treatment, a design theory can be formed, constituting reasonably justified generalizations about the effects of the designed artifact and its interaction with the actual problem context. If the design theory does indicate that the treatment is insufficient, it usually leads to further iterations of the design cycle. Otherwise, the treatment can be tried out in the actual problem context, via treatment implementation and validation. Whenever something is unclear and new knowledge is needed for one of the tasks in the design cycle, a knowledge

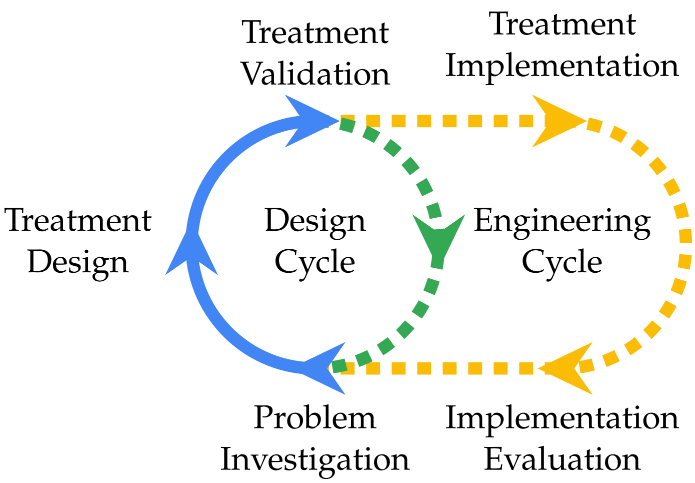


Fig. 1.1: Illustration of the three tasks of the Design Cycle as an inner loop of the Engineering Cycle. The treatment validation task should result in a Design Theory, based on which a decision can be made whether to take additional iterations of the Design Cycle or to proceed to implementation. In this ph.d. study, a single iteration of the design cycle has been completed as indicated by the solid blue line in the figure.

question is posed, that can aid in progressing the task. Opposite to technical research problems that ask for a change in the world and for which multiple solutions might exist, knowledge questions ask for knowledge about the past and present world. By nature, knowledge questions assume that there is only one correct answer. However, one needs to be aware that most answers to knowledge questions, especially empirical ones, might be subject to fallibilism. Nevertheless, the *"facts"* provided by these answers are indispensable e.g. to make the reasonably justified generalizations constituting the design theory.

As indicated by the blue solid arrow in Fig. 1.1, this study has been structured as a single iteration of the design cycle [4]. For each of the tasks in this iteration, one or more related knowledge questions have been formulated and answered. Within this ph.d. study, the purpose of the problem investigation and the treatment design was to either propose improvements for existing frameworks or to propose an entirely new framework. Again, since the technical research problem cannot reasonably be fully addressed, the ultimate goal of the treatment validation has been to develop a design theory predicting the advantages and disadvantages of using the proposed framework for implementing, sharing, and reusing parts of artificial robot cognition within the problem context of unstructured, dynamic, and uncertain environments.

To further focus the scope of this ph.d. study, models of the problem context have been developed with special emphasis on navigation and motion planning for mobile robots since it is believed that mobile robots covering large workspaces are especially prone to unstructured, dynamic and uncertain environments, and since there have already been proposed many great robot applications in human environments that require a mobile platform with these skills e.g. robotic-sales man, robot parcel delivery, robotic waitress, etc.

The remainder of Part I is devoted to giving a summary of how the different tasks in the decision cycle have been carried out and pointing out how the papers published and submitted during the study have contributed to the overall research objective of answering the technical research problem.

Chapter 2

Background and State-of-the-Art

2.1 Human-Robot Interaction

Since the required interaction with humans is deemed one of the main causes making it hard to develop robotic systems for environments outside of the traditional industrial environments, it is natural to ask the following knowledge question as part of the problem investigation.

Knowledge Question 1:

Which methodologies have previously been used to make robots interact with humans, and why have these methodologies not had more success?

In this study, this knowledge question has been sought answered through a literature review with the main findings summarized within this section.

The interaction between robots and humans is the topic of the rapidly growing research area of human-robot interaction (HRI) within the broader robotics community. As the name implies HRI is the study of interactions between humans and robots. Despite not being fully up to date [2] gives a good overview of the broadness of this area of research. According to [2] the research in this area can be divided into "foundations" and "high-level competencies", which can be further divided into subcategories. As the name implies "foundations" are basic capabilities and modalities that are precursors to high-level competencies and successful interaction with humans. The high-level competencies are not independent, and some may depend on others; e.g. navigation and Human-aware motion planning depends on the intentional action competencies "theory of mind" and "communicating intent". Naturally, safety should be one of the main objectives when designing

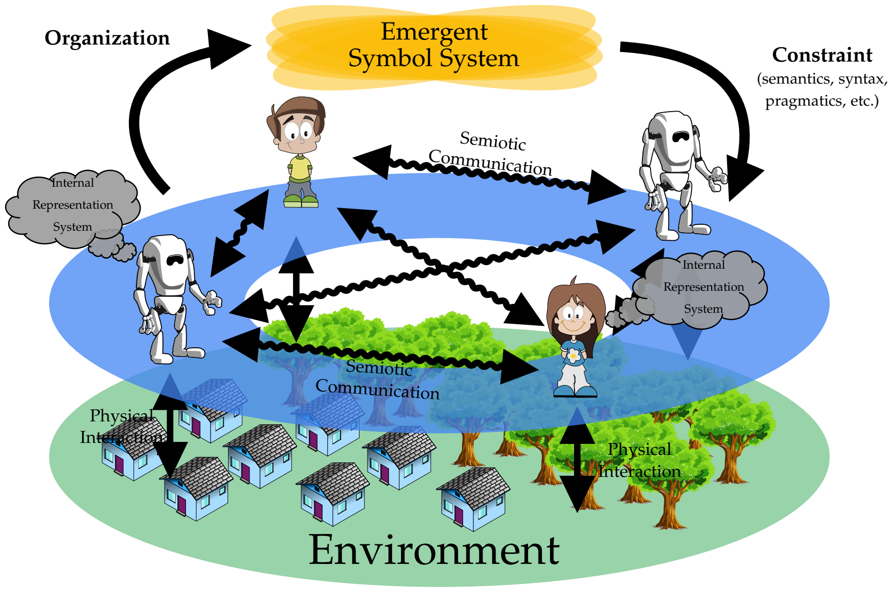


Fig. 2.1: Illustration of how an emergent symbol system emerges and evolves as the result of semiotic communication between cognitive agents in an environment. Through physical interaction with the environment, each agent can freely form an internal representation of that environment. However, if the agents want to communicate they must organize how things in the environment should be represented, at least for communication, by which the emergent symbol system emerges via consensus. If a strong consensus already exists in a society, then newcomers have to adapt to the already established emergent symbol system, and thus it puts some constraints on the newcomers' internal representation system. On the other hand, if the consensus is not strong, the emergent symbol system might drift.

a system for HRI. Therefore, different methods for safe HRI have been surveyed in [5]. [3] summarizes the insights that the researchers at the interAct laboratory at UC Berkeley *"have gained in integrating computational cognitive models of people into robotics planning and control"* using rational models of human behavior in a game-theoretical approach. Finally, methods and topics related to Human/social-aware navigation have been extensively surveyed in [6] and [7]. In [7] they conclude that

"a pluralism of algorithms with different starting points emerged, which rapidly enriches the respective literature. Even though important steps have been made, the effectiveness of (the) ***those works need to be further*** *improved and* ***standardize(d)*** *in order for robots to be accepted and co-exist in human populated environments in a daily fashion".*

What is evident from studying the above surveys is that the dominating methodology within HRI is to take a top-down approach in which robot applications are broken into ever smaller problems that are sought to be solved by incrementally changing highly specialized and non-standardized algorithms to produce small advancements without much regard for the overall system architecture.

In recent years, it has been pointed out by researchers within the newly formed research field of symbol emergence in robotics (SER) [8, 9], that such a top-down methodology is inappropriate for developing robots for real-world environments requiring interactions with humans. This is argued from the viewpoint that, whenever a robot is deployed into an environment with other cognitive agents it becomes a part of a dynamical system referred to as the emergent symbol system as illustrated in Fig. 2.1. The term emergent symbol system refers to the standardization of the way cognitive agents communicate within a certain society, i.e., the set of "rules" governing semiotic communication. If a robot does not understand and follow these "rules" it cannot communicate with the other cognitive agents in the environment. This set of "rules" is not given a priori but rather emerges and evolves dynamically through social interaction. Therefore, robots need to be able to adapt to and evolve with this emergent symbol system to efficiently and sustainably interact with humans in the long term. It is further argued that this can only be achieved using a bottom-up methodology and by architectures *"using general, not specifically tailored, mechanisms that provide an overarching framework for cognitive development"* [9].

In conclusion to knowledge question 1, a top-down methodology is commonly used for research in human-robot interaction and has provided many great results for well-defined and bounded application areas. However, this top-down methodology results in systems having limited learning and adaptive capabilities. As a consequence, robotic systems for unstructured and dynamic environments requiring efficient interaction with humans should predominantly be built in a bottom-up fashion from architectures providing general mechanisms. Furthermore, it has been pointed out that work within this field needs to be standardized.

2.2 Cognitive Architectures

Based on the conclusion reached through the literature review summarized in Section 2.1, the next pressing question is:

Knowledge Question 2:

What efforts have previously been done to construct architectures with general mechanisms for cognitive development?

To answer this knowledge question a literature review of the research field of cognitive architectures was found highly relevant. The research in cognitive architectures (CA) is centered around creating *"a concrete implementable*

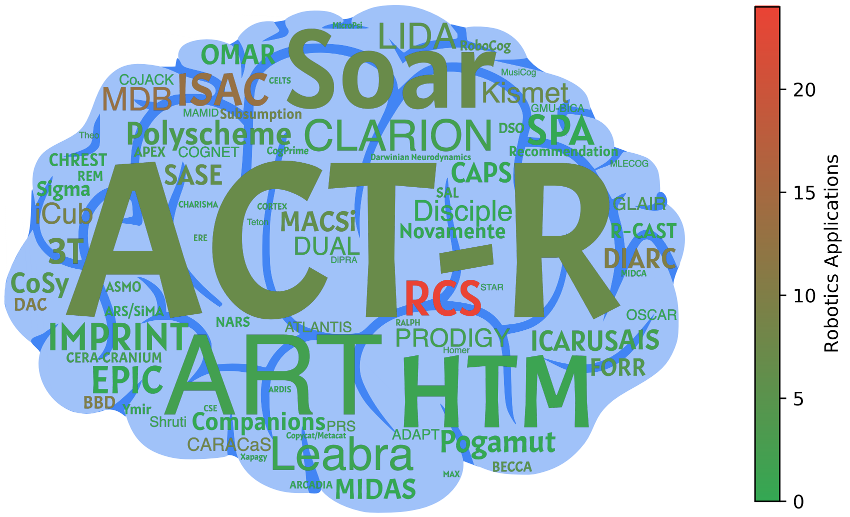


Fig. 2.2: Wordcloud based on data for the 84 cognitive architectures surveyed in [10]. Size indicates the total number of applications that each cognitive architecture has been used for. The color indicates the number of robotics-related applications that each Architecture has been used for.

model of the fixed structure that defines a mind" [11] with much of the research effort put in *"the core cognitive abilities, such as perception, attention mechanisms, action selection, memory, learning, reasoning, and meta reasoning"* [12]. The research in cognitive architectures dates back to the beginning of the 1970s, and it is estimated that there currently exist approximately three hundred different architectures with the most prominent being [12]: ACT-R [13], Soar [14, 15], CLARION [16], ICARUS [17], EPIC [18], and LIDA [19]. The practical application of the different cognitive architectures varies greatly from psychological experiments to robotics. Despite this, only a few architectures have put substantial efforts into applications to robotics as illustrated by Fig. 2.2. Nevertheless, as argued in [20] and [21] the general ideas behind cognitive architectures still potentially allow for complex autonomous robotic behavior. This is demonstrated e.g. by a fully Autonomous robotic salesman [22], a mobile robot assisting physicians in making geriatric assessments [23], and a robot performing advanced pick-and-place actions [24].

The structural organization and methods used to model core cognitive abilities vary just as much as the application areas. According to [12] architectures can roughly be categorized as either emergent (connectionist), symbolic (cognitivist) or hybrid hereof. In [12] it was further concluded that *"most of the newly developed architectures are hybrid"*, and that this category is *"showing* *the tendency to grow even more"*. In [25] a short introduction to the structure and functioning of 26 different cognitive architectures is given.

In conclusion to knowledge question 2, a lot of effort has been put into developing cognitive architectures, and it seems that hybrid approaches combining both emergent and symbolic approaches are gaining traction. As such cognitive architectures seem to be a good source of inspiration for artificial robot cognition. However, even though a few studies have shown promising results from utilizing cognitive architectures for robotics, cognitive architectures still do not seem to be widely utilized within the robotics community. As we elaborate upon in Paper A, this might again be due to a lack of standardization of the way models are implemented. This concluded the main problem investigation of this project, and the treatment design was initiated with the next knowledge question.

Knowledge Question 3:

What attempts have previously been made towards a unifying and standardized framework for implementing, sharing and reusing parts of artificial (robot) cognition?

In [26] the two foundational and prominent cognitive architectures ACT-R and Soar, and a relatively new cognitive architecture called Sigma [27] *"based partly on lessons learned from the two others"* are compared, and a standard model trying to capture a community consensus is proposed. In support of the observation made in [12], this standard model advocates the need for a hybrid combination of symbolic and statistical processing. Even though some consensus about the structure and functioning of human-like minds might exist, "the Standard Model of the Mind" is not complete, and strong consensus does not exist for all aspects of it as expatiated on in [11]. But more importantly, it does not advocate any practical way of implementing computationally feasible models, nor does it specify how parts of models can easily be shared and reused. Furthermore, most architectures such as ACT-R and Soar are based on a diverse set of specialized modules without an underlying theoretical coherent substrate. As a result, these architectures lack functional elegance making it hard to understand, use, reuse, maintain, and establish theoretical claims about them. The lack of functional elegance in other architectures is one of the main motivations behind the cognitive architecture Sigma, which also defines grand unification, generic cognition, and sufficient efficiency as desiderates. As we elaborate in Paper A, these desiderates are also highly relevant for a unifying and standardized framework for robot artificial cognition. Despite the ambitious desiderates of Sigma, the current implementation of the architecture does only support learning to a limited extent, does not efficiently represent continuous signals, and does not scale well. Similarly, it was also pointed out in [9] that the adaptability and developmental nature of many types of cognitive architectures, e.g., ACT-R,

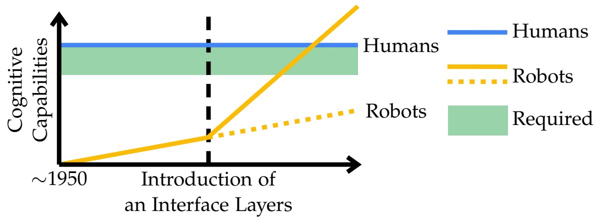


Fig. 2.3: Illustration of the possible benefits of introducing an interface layer for developing cognitive capabilities for robots. Before the introduction of a prober interface layer, the improvements to cognitive capabilities happen at one rate. If progress continues at this rate it might take several decades before the cognitive capabilities of robots reach the level required for unstructured, dynamic, and uncertain environments (the green area). With the introduction of an interface layer, the rate of progress would possibly increase immensely, and the required level of cognitive capabilities can be reached much earlier.

SOAR, and Clarion, are so limited that they would not allow robots to function as part of an emergent symbol system. As a remedy for this the authors further conclude that:

"Finding an appropriate way to integrate probabilistic generative models and neuro-dynamics models is crucial for developing computational cognitive architecture modeling symbol emergence in cognitive developmental systems" [9].

Some work has already been done in this direction by the SERKET [28] and Neuro-SERKET [29] frameworks. However, as we argue in Paper A these are not suitable for the unifying and standardized framework that is sought in this study. Mainly because other researchers advocate for the combinations of first-order logic and probabilistic graphical models [30], which is not supported by the two SERKET frameworks. At this point, one might question the necessity of combining both probabilistic, logical, and neuro-dynamics-inspired models. However, each type of model has its advantages. Logical models are usually easier for humans to understand, neuro-dynamics inspired models are great for correcting epistemic uncertainties, i.e., model errors, via their learning capabilities, while probabilistic models are great for modeling aleatoric uncertainties, i.e., the type of uncertainty that cannot simply be removed by incorporating more information into a model. As such, none of these types of models seems indispensable for the unifying and standardized framework sought in this study. Another cardinal conclusion from [9], is that

"developing an architecture that is decomposable and comprehensible while providing consistent learning results is important" [9].

This is concluded simply from the observation that artificial cognition is usually constructed from many components that need to be integrated into a conjoint architecture capable of system-wide learning. Although not pointed out in [9], decomposable and comprehensible architectures would provide the additional benefit of inherently making cooperation easier because they allow researchers and practitioners to share and reuse components rather than full architectures.

In addition to the above, the authors of [30] firmly argue that the overarching research field of artificial intelligence would benefit immensely from the introduction of one or more widely accepted interface layers. This is because anecdotal evidence has shown that such interface layers have triggered a period of rapid progress after their introduction in other subfields of computer science. It is argued that rapid progress is enabled because *"an interface layer separates innovation above and below it, while allowing each to benefit from the other"*. By deduction, with such interface layers, robotic systems based on cognitive architectures will be more likely to soon reach the level of cognitive capabilities required for unstructured, dynamic, and uncertain environments as illustrated in Fig. 2.3.

In conclusion to knowledge question 3, even though important steps have been done towards defining unifying frameworks from different combinations of probabilistic, logical, and neuro-dynamics models, all these attempts lack some of the components that are argued to be highly important by other researchers.

2.3 Requirements Specification

From the preceding analysis, a series of requirements for a framework providing unifying standards for implementing artificial robot cognition was derived and chosen. These requirements together with their contributing arguments are given below.

Treatment Requirements:

R1 Should embrace emergent (connectionist), symbolic (cognitivist) and hybrid approaches.

- If the framework cannot embrace emergent, symbolic and hybrid approaches, it will not be able to provide unifying standards for implementing artificial robot cognition.

R2 Should accommodate the combination of probabilistic, logical, and neuro-dynamics-inspired models.

- If the framework cannot accommodate the combination of probabilistic, logical, and neuro-dynamics-inspired models, it will not be suitable for the problem context of unstructured and dynamic environments perfused by uncertainty from human interaction.

R3 Should promote grand unification.

- If the framework cannot support grand unification, i.e., spanning all of cognition, there would be parts of cognition that the framework cannot support and thus it would not be able to provide unifying standards for implementing artificial robot cognition.

R4 Should promote generic cognition.
- If the framework cannot support generic cognition, i.e., spanning natural and artificial cognition, it will be harder to implement artificial robot cognition with characteristics of human cognition, since inspiration cannot be taken from implementations of artificial natural cognition.

R5 Should promote functional elegance.
- If the framework cannot support functional elegance, it will not be easy to share and reuse parts of artificial robot cognition because these parts will not be developed from a common reference point and thus harder to understand.

R6 Should promote sufficient efficiency.
- If the framework cannot support sufficient efficiency, it will not be suitable for developing robots for real-world environments.

R7 Should have at least one interface layer.
- If the framework can provide at least one interface layer, it will potentially speed up the development of artificial robot cognition and make development and reuse easier.

R8 Should support the Standard Model of the Mind.
- If the framework cannot support the Standard Model of the Mind, development within the scope of the framework will not be able to benefit from the community consensus that has begun to emerge.

R9 Should support decomposable implementations of artificial cognition allowing for system-wide learning.
- Decomposable implementations make cooperation easier because they allow researchers and practitioners to share and reuse components rather than full architectures.

None of the existing frameworks described in the preceding analysis fully satisfies all these requirements. Therefore, with this specification of treatment requirements, the design of a new framework was initiated. This is the topic of the next Chapter.

Chapter 3

Preview and Contributions

With the outset in the requirements specification derived in Chapter 2, this section presents how each of the papers composed as part of this ph.d. study has contributed to the treatment design and treatment validation. For each paper, a short summary is given highlighting their specific contributions to state-of-the-art. The papers can be found in their full length in Part II.

3.1 Paper A - Treatment Design

Mini-Review

Paper A

M. R. Damgaard, R. Pedersen, and T. Bak, "Toward an idiomatic framework for cognitive robotics," *Patterns*, vol. 3, no. 7, p. 100533, 2022. [Online]. Available: `https://doi.org/10.1016/j.patter.2022.100533`

Paper A identifies a potential need for a unifying and standardized framework for developing cognitive architectures aimed at cognitive robotics. Therefore, the Generalized Cognitive Hourglass model centered around probabilistic programs is proposed as such a framework. By dividing the development of cognitive architectures into a series of layers the framework embraces the same four desiderates as Sigma, and provides an interface layer between models of cognition and the algorithms that implement them on physical computational devices. By grounding the framework in probabilistic programming, the proposed framework makes it possible to combine probabilistic, logical, and neuro-dynamics-inspired models, thus embracing emergent, symbolic and hybrid approaches. Besides proposing a novel framework, Paper A also reviews topics essential for the proposed framework such as probabilistic programming and probabilistic programming languages. Furthermore, Paper A formally introduces both a new graphical representation entitled "Generative Flow Graphs" and the new inference approach entitled "Stochastic message-passing". Finally, Paper A exemplifies key ideas of the framework by summarizing its relation to two other studies.

Contribution to the Research Objective

As should be clear from the mini-review, Paper A is essential to the overall research objective of this ph.d. study as it proposes a concrete design of an artifact for treating the problem context in the Technical Research Problem specified in Section 1.2 while satisfying the treatment requirements put forward in Section 2.2. However, this design proposal cannot stand alone without a validation of the treatment. Therefore, the remainder of this chapter is devoted to the last task in the design cycle, i.e., treatment validation. Treatment validation is essentially the process of investigating the interaction between a model of the problem context and a prototype of the proposed artifact to justify that the treatment satisfies the specified requirements [4]. Within this ph.d. study, any algorithm or system developed within the scope of the proposed framework can be considered a prototypical usage of the proposed artifact, i.e. the proposed framework. Papers A, B, C, and D each contribute to the treatment validation by investigating different related knowledge questions.

First and foremost, Paper A contributes to the treatment validation via a conceptual analysis justifying the proposed framework, which is backed by a discussion of how the two SERKET frameworks, the cognitive architectures Sigma, and the system called Alchemy can all be viewed as special cases of the framework. Paper A furthermore empirically considers the following knowledge questions related to requirements **R5** and **R7**.

Knowledge Question 4:

Can probabilistic programs provide an appropriate interface layer and foundation for functional elegance?

In Section A.8 of Paper A, knowledge question 4 is approached empirically through analysis of the experience of implementing two specific robot applications utilizing the same probabilistic programming idiom. Based on this analysis it is concluded that *"it is possible to develop generally applicable models of cognition that can easily be adapted and/or extended to new use cases"* and *"we do not need to re-implement the idiom itself to accommodate this*(another) *inference algorithm"*. These conclusions, further substantiate that probabilistic programs can indeed provide an appropriate interface layer and foundation for functional elegance.

3.2 Paper B - Treatment Validation 1

Mini-Review

Paper B
M. R. Damgaard, R. Pedersen, and T. Bak, "Study of variational inference for flexible distributed probabilistic robotics," *Robotics*, vol. 11, no. 2, pp. 38:1–38:19, 2022, article in a periodical. [Online]. Available: `https://doi.org/10.3390/robotics11020038`

Paper B summarizes three major approaches to solving probabilistic inference problems from which it is concluded that variational inference can be viewed as a compromise between computational efficiency and flexibility. Paper B then proposes the combined usage of stochastic variational inference and message-passing as a flexible tool for solving inference problems in probabilistic robotics in a distributed manner. The approach can be seen as a precursor to the inference approach entitled "Stochastic message-passing" in Paper A. In Paper B, the approach is exemplified by deriving an algorithm for multi-robot navigation with cooperative avoidance under uncertainty. Through simulations of the multi-robot navigation problem, it is shown that the approach can outperform the problem-specific algorithm called B-UAVC. Finally, a real-world single-case mechanism experiment with two robots validates that the approach can be computationally tractable for online implementation.

Contribution to the Research Objective

Although it may not be immediately clear from Paper B, the underlying purpose of the presented study within the overarching ph.d. study is to get an initial answer to the following knowledge question related to the validation of requirement **R6**.

Knowledge Question 5:

Can artificial robot cognition be developed within the scope of the proposed framework provide sufficient efficiency for real-world robotics applications in unstructured, dynamic, and uncertain environments?

Paper B approaches this knowledge question empirically from one simulation-based and one real-world single-case mechanism experiment. At a principal level knowledge question 5 boils down to the possibility of providing artificial robot cognition that is computationally tractable for online implementation based on inference in probabilistic programs. To investigate this, the problem of navigation with cooperative avoidance under uncertainty was chosen as a model of the problem context, i.e., unstructured, dynamic, and uncertain environments, and a relatively simple probabilistic program was implemented, acting as a prototype for the type of models developed within the scope of the proposed framework. Based on these experiments it is concluded: *"that sufficient computational efficiency is possible on standard hardware"*. From this, it cannot be concluded that the framework provides sufficient efficiency in all situations, however, it indicates plausibility of it. Paper B also addresses the following interesting and relevant knowledge questions.

Knowledge Question 6:

How do algorithms developed within the scope of the proposed framework perform in unstructured, dynamic, and uncertain environments compared to problem-specific algorithms?

This question is of interest because it appends to the justification of the proposed framework. In other words, if the methods prescribed by the framework cannot provide performance comparable to other methods, then the framework should perhaps be reconsidered. In Paper B, this knowledge question is also addressed empirically via the simulation-based single-case mechanism experiment. From this, it is concluded that a specific algorithm developed within the scope of the proposed framework *"performs as well as, if not better than, B-UAVC specifically made for the problem of multi-robot collision avoidance"*. Based on this, there does not seem to be a reason to reconsider the framework or the methods prescribed by it. Finally, Paper B also investigates knowledge question 7 which is related to requirement **R9**.

Knowledge Question 7:

Can the proposed framework support decomposable implementations of artificial cognition?

Both experiments presented in Paper B demonstrate that inference within probabilistic programs cannot only support decomposable implementations but also distributed and decentralized implementations via the combination of stochastic variational inference and message-passing. Since inference within probabilistic programs has been chosen to be at the heart of the proposed framework, it can thus be concluded that the proposed framework supports decomposable implementations to the extent that models of artificial cognition can be decomposed into variational inference sub-problems.

3.3 Paper C - Treatment Validation 2

Mini-Review

Paper C
M. R. Damgaard, R. Pedersen, and T. Bak, "A probabilistic programming idiom for active knowledge search," in *2022 International Joint Conference on Neural Networks (IJCNN)*, July 2022, pp. 1–9. [Online]. Available: `https://www.doi.org/10.1109/IJCNN55064.2022.9892094`

Paper C proposes a probabilistic programming idiom for active knowledge search based on the memory structure of the Standard Model of the Mind. The probabilistic programming idiom defines a decision model that somewhat resembles other decision models such as the Partially observable Markov decision process. However, the proposed idiom uses special general-purpose decision variables to guide action selection. To exemplify the usage of the proposed idiom, an algorithm for active mapping and robot exploration is derived from the idiom. To evaluate the qualities of the derived algorithm an extensive simulation study is presented utilizing the House-Expo dataset within a modified version of the PseudoSLAM simulator. From this simulation study, it is concluded that the idiom works as expected, but also that it would benefit from additional functionality.

Contribution to the Research Objective

Within the scope of the overarching ph.d. study, the purpose of Paper C is to provide empirical evidence for evaluating the following knowledge question related to requirement **R8**.

Knowledge Question 8:

Can the proposed framework support the Standard Model of the Mind?

To obtain this empirical evidence it was once again decided to do a simulation-based single-case mechanism experiment. For this experiment, it was decided to concentrate on the "unstructured" and "uncertain" parts of the problem context to keep things as simple as possible. Therefore, the model of the environment was kept static. Furthermore, since a strong consensus does not exist for all aspects of the Standard Model of the Mind, it was decided to concentrate on a part of the Standard Model of the Mind backed by strong consensus. Therefore, the prototype used for this experiment took the outset in the memory structure advocated by the Standard Model of the Mind. Based on the results of the simulation-based single-case mechanism experiment it can be concluded that the proposed framework at least allows the incorporation of the memory structure of the Standard Model into a simple decision model.

3.4 Paper D - Treatment Validation 3

Mini-Review

Paper D

M. R. Damgaard, R. Pedersen, and T. Bak, “Escaping local minima via appraisal driven responses,” *Preprints*, 2022110511, 2022. [Online]. Available: `https://doi.org/10.20944/preprints202211.0511.v1`

Paper D takes the outset in the limitations identified for the probabilistic programming idiom proposed in Paper C. By taking inspiration from the reflective and deliberative control mechanisms used in cognitive architectures such as SOAR and Sigma, Paper D proposes an alternative decision mechanism driven by architectural appraisals. This decision mechanism is implemented as generally applicable probabilistic programming idioms extending upon the probabilistic programming idiom proposed in Paper C. By providing automatic context-dependent switching between exploration-oriented, goal-oriented, and backtracking behavior, the proposed probabilistic programming idiom enables robots to overcome some types of impasses. To test the effectiveness of the proposed probabilistic programming idiom two simulations studies were performed focusing on the exploration-oriented and the goal-oriented capabilities, respectively. From the simulation studies focusing on the exploration-oriented capabilities, it is concluded that the new decision mechanism indeed performs better than the one proposed in Paper C. From the simulation studies focusing on goal-oriented capabilities, it is furthermore concluded that the new decision mechanism in some cases can outperform State-of-the-art goal-oriented algorithms derived from problem-specific knowledge. Finally, Paper D offers a detailed discussion of the computational time of the proposed probabilistic programming idiom. From this discussion, it is concluded that optimizations of the implementation might be necessary.

Contribution to the Research Objective

As Paper D builds upon concepts from both Papers B and C, the main purpose of it was to further substantiate some of the conclusions that have already been reached. However, this time based on a more complicated prototypical usage of the framework. As in Papers B and C, the empirical evidence in Paper D was also obtained through simulation-based single-case mechanism experiments. This was primarily done to obtain results that can be compared to our previous research in Paper C as well as research on problem-specific methods. From these experiments, Knowledge Question 6 can be revised. Even though the proposed decision mechanism does not use any problem-specific knowledge, the second simulation study within Paper D shows that the proposed decision mechanism in many situations performs as well as and even better than state-of-the-art methods incorporating problem-specific knowledge. As a result, based on the application metric this experiment neither indicates any reason to reconsider the framework nor the methods prescribed by it.

As the approach proposed in Paper D is implemented as a series of probabilistic programming idioms mostly building on top of each other hierarchically, Paper D also demonstrates how decomposable implementations of artificial cognition can be implemented within the proposed framework. Thereby, providing additional evidence to substantiate the previous answer to knowledge question 7 related to requirement **R9**. Related to knowledge Question 4 and requirement **R5**, the success of the simulations in Paper D also provides positive evidence that probabilistic programs can be used as a foundation for functional elegance since all of the idioms were implemented as probabilistic programs.

Furthermore, as the decision mechanism proposed in Paper D builds on the decision mechanism proposed in Paper C, the success of the simulations in Paper D provides additional positive evidence that the proposed framework allows the incorporation of the memory structure of the Standard Model into decision models. Thereby, substantiating the previous answer to Knowledge Question 8 related to requirement **R8**.

Finally, with the discussion of the computational time of the decision mechanism, Paper D also provides evidence for answering Knowledge Question 5 related to requirement **R6**. This evidence is however a bit more ambiguous. On one hand, it is concluded that the current implementation of the decision mechanism does not match the time scales at which humans can deliver similar responses. This could of course be seen as negative evidence for the possibility of providing sufficient efficiency within the framework. On the other hand, it is also argued in Paper D that further optimization of the implementation and additional functionality might make up for the observed incongruity. In addition to this, it might not be necessary to achieve human-level performance to support the anticipated uses in real-time. As such, the results are neither enough to discard the possibility that the framework can provide sufficient efficiency for more complex implementations of decision-making. Thus, for now, this yields a conditional result concerning the general possibility of achieving sufficient efficiency for more complex implementations of decision-making within the proposed framework.

3.5 Software

For Papers B, C and D software had to be implemented before the simulation-based and real-world single-case mechanism experiments could be conducted. Following the approach prescribed by the proposed framework, the main functionalities used in each of these papers have been implemented as generally applicable probabilistic programming idioms in the form of abstract python classes. By inheriting from these abstract classes and implementing a few abstract methods, it should in principle be easy for other researchers and practitioners to reuse the main functionalities developed as part of this study. All of these functionalities together with each of the simulations have been collected into a GitHub repository available at [31]. The specific versions of this repository used for each paper can be found at

- [32] for Paper B,
- [33] for Paper C,
- and [34] for Paper D.

For the real-world single-case mechanism experiments in Paper B, a ROS2 package was developed to make communication between multiple robots possible. This package can be found at [35]. The repository at [31] provides evidence for Knowledge Question 9 related to requirements **R5**, **R7** and **R9**

Knowledge Question 9:
Can the proposed framework support reusable implementations of artificial cognition?

In the version of the repository availeable at the time that this disseration was written [34], all of the simulations conducted in Papers B, C and D have been implemented with a basis in the same limited number of abstract python classes. Thus, this repository demonstrates how implementations of artificial cognition developed within the scope of the proposed framework can be reused for different applications. The repository thereby provides positive evidence for knowledge question 9. However, it cannot be concluded *how easy* it is for others to reuse such implementations because a proper judgment of this would require other researchers and practitioners to actually reuse the functionalities and provide feedback on this process, which at the time of writing has not been the case.

3.6 Design Theory

In summary of the treatment validation described in previous sections, Papers A through D have provided empirical evidence that the framework can support functional elegance (**R5**), can yield sufficient efficiency for simple models of decision (**R6**), provides a useful interface layer (**R7**), at least to some extent supports the Standard Model of the Mind (**R8**), and does allow decomposable implementations of artificial cognition (**R9**). These properties have been validated for cognitive functionalities related to decision-making in models of the problem context that are unstructured, dynamic and uncertain due to the following assumptions.

Environments Assumptions:

- Unstructured since no problem-specific information is allowed to/can be exploited.
- Dynamic due to other robots moving around that communicate their intentions explicitly.
- Uncertain since Sensor measurements are noisy and motion models are imperfect.

Besides the conceptual analysis provided in Paper A, the presented treatment validation does not touch upon requirements **R1** through **R4**, and as such is only partial. However, some valuable knowledge can still be extracted about the properties of the proposed framework. More specifically, based on the treatment validation the following theory can be formed.

Design Theory:
A framework designed like the one proposed in Paper A utilized in relation to environments that are unstructured, dynamic and uncertain due to the reasons given by the above Environments Assumptions can provide implementations of artificial robot cognition related to decision-making. Implementations like these

- *can be decomposable and reused,*
- *can utilize distributed and decentralized computations,*
- *can support the memory structure of the Standard Model of the Mind,*
- *can yield sufficient efficiency for simple models of decision similar to those used in Paper B, and presumably for more complex models as well,*
- *shows that probabilistic programs provide a useful interface layer,*
- *and shows that probabilistic programs provide a foundation for functional elegance.*

The above design theory gives a lot of useful insights, however, it is also limited to some extent. The design theory is limited in the sense that it only considers implementations of decision-making and no other parts of artificial robot cognition. It is also relatively limited in the types of environments that it considers. However, these limitations are only a result of a limited treatment validation from which no broader generalization can reasonably be made, and it is especially noteworthy that no major deficiencies that could rule out the proposed framework design have been identified. In conclusion, this design theory is not mature enough to predict the outcome of using the proposed framework to develop full artificial robot cognition for use in unstructured and dynamic environments perfused by uncertainty from human interaction. Further treatment validation will have to be done, to mature the design theory to the required extent.

Chapter 4

Conclusion and Outlook

4.1 Conclusion

This ph.d. study took its outset in a hypothesis that naturally led to the technical research problem in Section 1.2. By structuring the study via the Design Cycle, the main contribution of this study, as given below, has been achieved.

Main Contribution:

The main contribution of this study has been the proposal of a design of a new framework that could potentially provide unifying standards for implementing artificial robot cognition with characteristics of human cognition that can easily be shared and reused.

With the design theory formed in Section 3.6, important steps toward answering the technical research problem have been taken. The design theory furthermore shows that the proposed framework offers many great properties. The design theory, however, also has some limitations that can only be ameliorated via further treatment validation. In addition to the main contribution, this study has also resulted in other noteworthy derivative contributions, e.g.,

Derivative Contributions:

1. a novel graphical representation of probabilistic programs called "generative flow graphs" that can ease the dissemination of new models of parts of cognition developed within the proposed framework (Paper A),

2. a novel approach for combining message-passing and stochastic variational inference called "Stochastic Message-Passing" (Paper A and Paper B),

3. an algorithm for multi-robot navigation with cooperative avoidance under uncertainty utilizing "Stochastic Message-Passing" that can perform as well as, if not better than, a state-of-the-art algorithm (Paper B),
4. a probabilistic programming idiom for the problem of acquiring new knowledge about an environment (Paper C),
5. an algorithm for the specific problem of active mapping and robot exploration based on this idiom (Paper C),
6. and a novel decision mechanism driven by architectural appraisals that provide automatic context-dependent switching between exploration-oriented, goal-oriented, and backtracking behavior, allowing a robot to overcome impasses (Paper D).

So, is the time ripe for developing commercial service robots that assist people in everyday life? In the author's opinion, a lot of work still has to be done to unify the way artificial robot cognition is implemented before roboticists can overcome the immense task of developing robots that can conquer unstructured and dynamic environments perfused by uncertainty from human interaction. The hope is that the proposed framework will be the stone that creates sufficient ripples allowing researchers and practitioners to overcome this immense task.

4.2 Suggestions for Future Work

As stated in Section 3.6, the treatment validation carried out in this study has been limited to decision-making and with respect to the types of environments considered. To obtain a design theory with a broader scope, it will be necessary to carry out treatment validation with a focus on other parts of artificial cognition such as perception, general learning capabilities, the formation of knowledge and memories, comprehension and production of language, and general problem-solving. Further treatment validation would also have to consider robots that not only are capable of moving around in their environments but also are capable of physically manipulating their environment, e.g., via end effectors mounted on arms. Similarly, further treatment validation would also have to consider environments that are dynamic due to the interaction with humans, rather than other robots simply moving around. Especially it is important to consider cases where other cognitive agents do not communicate their intention or in any other way do not collaborate with the robots. All of the above will make it possible to broaden the scope of the design theory, and possibly expose limitations of the proposed framework that is currently unknown.

References

[1] W. Burgard, A. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "The interactive museum tour-guide robot," 01 1998, pp. 11–18.

[2] A. Thomaz, G. Hoffman, and M. Cakmak, *Computational Human-Robot Interaction*. now, 2016. [Online]. Available: https://ieeexplore.ieee.org/document/8186865

[3] A. D. Dragan, "Robot planning with mathematical models of human state and action," *CoRR*, vol. abs/1705.04226, 2017. [Online]. Available: http://arxiv.org/abs/1705.04226

[4] R. J. Wieringa, *Design Science Methodology for Information Systems and Software Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014. [Online]. Available: https://doi.org/10.1007/978-3-662-43839-8_10

[5] P. A. Lasota, T. Song, and J. A. Shah, *A Survey of Methods for Safe Human-Robot Interaction*. now, 2017. [Online]. Available: https://ieeexplore.ieee.org/document/8186877

[6] J. Rios-Martinez, A. Spalanzani, and C. Laugier, "From proxemics theory to socially-aware navigation: A survey," *International Journal of Social Robotics*, vol. 7, no. 2, pp. 137–153, Apr 2015. [Online]. Available: https://doi.org/10.1007/s12369-014-0251-1

[7] K. Charalampous, I. Kostavelis, and A. Gasteratos, "Recent trends in social aware robot navigation: A survey," *Robotics and Autonomous Systems*, vol. 93, pp. 85 – 104, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889016302287

[8] T. Taniguchi, T. Nagai, T. Nakamura, N. Iwahashi, T. Ogata, and H. Asoh, "Symbol emergence in robotics: a survey," *Advanced Robotics*, vol. 30, no. 11-12, pp. 706–728, 2016. [Online]. Available: https://doi.org/10.1080/01691864.2016.1164622

[9] T. Taniguchi, E. Ugur, M. Hoffmann, L. Jamone, T. Nagai, B. Rosman, T. Matsuka, N. Iwahashi, E. Oztop, J. Piater, and F. Wörgötter, "Symbol emergence in cognitive developmental systems: A survey," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 11, no. 4, pp. 494–516, 2019.

[10] I. Kotseruba and J. K. Tsotsos, "40 years of cognitive architectures: core cognitive abilities and practical applications," *Artificial Intelligence Review*, vol. 53, no. 1, pp. 17–94, 2020, article in a periodical. [Online]. Available: https://doi.org/10.1007/s10462-018-9646-y

[11] P. S. Rosenbloom, "Lessons from mapping sigma onto the standard model of the mind: Self-monitoring, memory/learning, and symbols," in *AAAI Fall Symposia*, 2017. [Online]. Available: http://bcf.usc.edu/~rosenblo/Pubs/SM%20Symp%20Sigma%202017%20Revised%20D.pdf

[12] I. Kotseruba and J. K. Tsotsos, "40 years of cognitive architectures: core cognitive abilities and practical applications," *Artificial Intelligence Review*, Jul 2018. [Online]. Available: https://doi.org/10.1007/s10462-018-9646-y

[13] F. E. Ritter, F. Tehranchi, and J. D. Oury, "Act-r: A cognitive architecture for modeling cognition," *Wiley Interdisciplinary Reviews: Cognitive Science*, vol. 10, no. 3, p. e1488, 2019.

[14] J. E. Laird, A. Newell, and P. S. Rosenbloom, "Soar: An architecture for general intelligence," *Artificial Intelligence*, vol. 33, no. 1, pp. 1 – 64, 1987. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0004370287900506

[15] J. E. Laird, *The Soar Cognitive Architecture*. The MIT Press, 2012.

[16] R. Sun, *The CLARION Cognitive Architecture: Extending Cognitive Modeling to Social Simulation*. Cambridge University Press, 2005, p. 79–100.

[17] D. Choi and P. Langley, "Evolution of the icarus cognitive architecture," *Cognitive Systems Research*, vol. 48, pp. 25 – 38, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389041716302182

[18] D. E. Kieras and D. E. Meyer, "An overview of the epic architecture for cognition and performance with application to human-computer interaction," *Human–Computer Interaction*, vol. 12, no. 4, pp. 391–438, 1997. [Online]. Available: https://doi.org/10.1207/s15327051hci1204_4

[19] S. Franklin, T. Madl, S. Strain, U. Faghihi, D. Dong, S. Kugele, J. Snaider, P. Agrawal, and S. Chen, "A lida cognitive model tutorial," *Biologically Inspired Cognitive Architectures*, vol. 16, pp. 105 – 130, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2212683X16300196

[20] T. D. Kelley and C. Lebiere, "From cognitive modeling to robotics: How research on human cognition and computational cognitive architectures can be applied to robotics problems," in *Advances in Human Factors in Simulation and Modeling*, D. N. Cassenti, Ed. Cham: Springer International Publishing, 2019, pp. 273–279.

[21] M. Scheutz, J. Harris, and P. Schermerhorn, "Systematic integration of cognitive and robotic architectures," *Advances in Cognitive Systems*, vol. 2, pp. 277–296, 2013.

[22] A. Romero-Garcés, L. V. Calderita, J. Martínez-Gómez, J. P. Bandera, R. Marfil, L. J. Manso, A. Bandera, and P. Bustos, "Testing a fully autonomous robotic salesman in real scenarios," in *2015 IEEE International Conference on Autonomous Robot Systems and Competitions*, April 2015, pp. 124–130.

[23] A. Bandera, J. P. Bandera, P. Bustos, L. V. Calderita, A. Duenas, F. Fernández, R. Fuentetaja, A. Garcıa-Olaya, F. J. Garcıa-Polo, J. C. González *et al.*, "Clarc: a robotic architecture for comprehensive geriatric assessment," in *Workshop on Physical Agents*, vol. 1, 2016, pp. 1–8.

[24] J. R. Wilson, E. Krause, M. Scheutz, and M. Rivers, "Analogical generalization of actions from single exemplars in a robotic architecture," in *Proceedings of the 2016 International Conference on Autonomous Agents &#38; Multiagent Systems*, ser. AAMAS '16. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2016, pp. 1015–1023. [Online]. Available: http://dl.acm.org/citation.cfm?id=2937029.2937073

[25] A. V. Samsonovich, "Toward a unified catalog of implemented cognitive architectures," in *Proceedings of the 2010 Conference on Biologically Inspired Cognitive Architectures 2010: Proceedings of the First Annual Meeting of the BICA Society*. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2010, pp. 195–244. [Online]. Available: http://dl.acm.org/citation.cfm?id=1893313.1893352

[26] J. Laird, C. Lebiere, and P. Rosenbloom, "A standard model of the mind: Toward a common computational framework across artificial intelligence, cognitive science, neuroscience, and robotics," *AI Magazine*, vol. 38, p. 13, 12 2017.

[27] P. S. Rosenbloom, A. Demski, and V. Ustun, "The sigma cognitive architecture and system: Towards functionally elegant grand unification," *Journal of Artificial General Intelligence*, vol. 7, no. 1, pp. 1–103, 2017, article in a periodical. [Online]. Available: https://doi.org/10.1515/jagi-2016-0001

[28] T. Nakamura, T. Nagai, and T. Taniguchi, "Serket: An architecture for connecting stochastic models to realize a large-scale cognitive model," *Frontiers in Neurorobotics*, vol. 12, pp. 25:1–25:16, 2018, article in a periodical. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnbot.2018.00025

[29] T. Taniguchi, T. Nakamura, M. Suzuki, R. Kuniyasu, K. Hayashi, A. Taniguchi, T. Horii, and T. Nagai, "Neuro-serket: Development of integrative cognitive system through the composition of deep probabilistic generative models," *New Gen. Comput.*, vol. 38, no. 1, p. 23–48, Mar. 2020, article in a periodical. [Online]. Available: https://doi.org/10.1007/s00354-019-00084-w

[30] P. Domingos and D. Lowd, *Markov Logic: An Interface Layer for Artificial Intelligence*, 1st ed. Morgan and Claypool Publishers, 2009, an entire book.

[31] M. R. Damgaard, "ProbMind," GitHub repository, https://github.com/damgaardmr/probMind, 2022, software available online.

[32] ——, "probmind - version used for paper b," 2022. [Online]. Available: https://github.com/damgaardmr/probMind/tree/d0ba27687b373ff04eb790ef38b21ca8572d8c8a/

[33] ——, "probmind - version used for paper c," 2022. [Online]. Available: https://github.com/damgaardmr/probMind/tree/paper_release

[34] ——, "probmind - version used for paper d," 2022. [Online]. Available: https://github.com/damgaardmr/probMind/tree/ec996e295575c384879b3d72cfc7e64b8085b9a5

[35] ——, "Variational inference navigation," https://github.com/damgaardmr/VI_Nav/tree/8af532f5e6618d46f3498460af6459e57261fc91, 2021.

Part II

Papers

Paper A

Toward an Idiomatic Framework for Cognitive Robotics

Malte Rørmose Damgaard, Rasmus Pedersen, and Thomas Bak

The paper has been published in
Patterns Volume 3, Issue 7, 100533, 2022.
`https://doi.org/10.1016/j.patter.2022.100533`

Bigger Picture

For many decades robots have been expected to transfigure the world we live in, and in many ways, they already have by increasingly taking over dull, dirty, and dangerous jobs. However, for robots to integrate fully and seamlessly into human societies, robots need to be able to learn and reason from experience and effectively deal with unpredictable and dynamic environments. Developing robotic systems with such intelligence is a tremendous and non-trivial task, which has led to the foundation of the new multidisciplinary scientific field called Cognitive Robotics, merging research in adaptive robotics, cognitive science, and artificial intelligence. To ease the merge of research from these scientific fields, we propose a general framework for developing intelligent robotic systems based on recent advancements in the machine learning community. Hopefully, this framework will aid researchers and practitioners in bringing even more helpful robots into our societies.

Abstract

Inspired by the "Cognitive Hourglass" model presented by the researchers behind the cognitive architecture called Sigma [1], we propose a framework for developing cognitive architectures for cognitive robotics. The purpose of the proposed framework is foremost to ease the development of cognitive architectures by encouraging cooperation and re-use of existing results. This is done by proposing a framework dividing the development of cognitive architectures into a series of layers that can be considered partly in isolation, and some of which directly relate to other research fields. Finally, we give introductions to and review some topics essential to the proposed framework. The paper also outlines a set of applications.

A.1 Introduction

Research in cognitive robotics originates from a need to perform and automate tasks in dynamic environments and in close or direct interaction with humans. Uncertainty about the environment and complexity of the tasks require robots with the ability to reason and plan while being reactive to changes in their environment. To achieve such behavior, robots cannot rely on predefined rules of behavior [2] and inspiration is taken from cognitive architectures.

Cognitive architectures provide a model for information processing that can capture robot functionalities. In combination with acquired sensory data, they can potentially generate intelligent autonomous behavior [3]. Cognitive architectures dates back to the 1950s [4] with a grand goal of implementing a full working cognitive system [1]. From this considerable challenge, an abundance of architectures have evolved, and a recent survey suggests that the number of existing architectures has reached several hundred [4]. Some are aimed towards robotics application, e.g., Robo-Soar [5], CARACaS [6], and RoboCog [7]. Unfortunately, most of these architectures take wildly different approaches to model cognition and are implemented in different programming languages. Furthermore, most of these architectures are constructed from a diverse set of specialized modules, making it non-trivial to expand upon, combine, and re-use parts of these architectures. In fact, this could be one of the contributing reasons for the abundance of architectures. The authors of a recent study of cognitive architectures related to the iCub robot explain their decision to start from scratch rather than relying on existing architectures was: "this decision was made in order to gain more freedom for future expansions of the architecture" [8]. In other words, despite the abundance of architectures, existing architectures were not deemed flexible enough to build upon. Following the arguments for developing an interface layer for artificial intelligence put forward by other researches [9], we argue that a unifying and standardized framework for developing new cognitive architectures aimed at cognitive robotics could potentially remedy these issues and ease the development of cognitive robotics.

In recent years a community consensus has emerged about a standard model of humanlike minds, i.e., computational entities whose structures and processes are substantially similar to those found in human cognition [10]. While this "Standard Model of the Mind" spans key aspects of structure and processing, memory and content, learning, and perception and motor, it is agnostic to the best practice for modeling and implementing these things [10].

In line with the idea proposed by the researchers behind the cognitive architecture Sigma [1], we argue that the evolution of the scientific field of cognitive robotics could benefit from anchoring new implementations around a common theoretical elegant base separating a specific model of a part of cognition from the algorithm that implements it. Furthermore, this theoretical base could allow new functionalities to evolve hierarchically just like software libraries build on top of each other. Thereby, allowing the discussions and development to flourish at different levels of abstractions, and allow synergy with other research fields.

To explain the cognitive architecture Sigma [1], the authors present a Cognitive Hourglass model based on the following four desiderata:

- Grand Unification, spanning all of cognition,
- Generic Cognition, spanning both natural and artificial cognition,
- Functional Elegance, achieving generically cognitive grand unification

with simplicity and theoretical elegance,

- Sufficient Efficiency, efficient enough to support the anticipated uses in real-time.

While Grand Unification and Sufficient Efficiency aligns well with the needs of cognitive robotics, the need for Generic Cognition and Functional Elegance is subtle for cognitive robotics. Although the end goal of cognitive robotics might only be functional artificial intelligence, building upon something that potentially is also able to model natural intelligence would allow artificial intelligence to more easily benefit from insights obtained by the modeling of natural intelligence and vice versa. Similarly, Functional Elegance is not a goal of cognitive robotics per se. Still, it could allow researchers and practitioners working on different levels of cognition to obtain a common reference point and understanding at a basic level, potentially easing co-operation and re-use of results and innovative ideas.

In an attempt to obtain all four of these desiderata, the so-called "graphical architecture" based on inference over probabilistic graphical models is placed at the waist of Sigma's Cognitive Hourglass model glueing everything together just like the Internet Protocol (IP) in the Internet-hourglass model [11]. Functional elegance is obtained by recognizing and developing general architectural fragments, and based on these defining idioms that can be re-used in modelling different parts of cognition. Having defined sufficiently general idioms, the hope is to be able to develop full models of cognition from a limited set of such idioms and thereby achieve functional elegance, while at the same time achieving the three other desiderata [1]. With roots in the given desiderata, Sigma's Cognitive Hourglass model, in many ways, could constitute a unifying and standardized framework for cognitive robotics. However, as we will elaborate on in Section A.2 the model commits to specific architectural decisions, which hinders the utilization of new technology and ideas. E.g., their commitment to the sum-product algorithm prevents the use of new algorithms for efficient probabilistic inference. The benefits of utilizing probabilistic graphical models specifically for cognitive robotics have recently been corroborated in a lot of studies. For example, learning and representing the hierarchical structure of concepts [12], simultaneous lexical and spatial concept acquisition [13], navigation utilizing the learned concepts [14], and the interaction between multiple probabilistic graphical models [15] has been studied. This research has led to two frameworks called SERKET [16] and its extension Neuro-SERKET [17] with the goal of connecting multiple probabilistic graphical models on a large scale to construct cognitive architectures for robotics. Being based solely on probabilistic graphical models, SERKET and Neuro-SERKET currently do not seem to incorporate logic, making it non-trivial to implement symbolic approaches in these frameworks. In fact, researchers behind the work related to Markov Logic and the system called

Alchemy have argued that the combination of logic, especially first-order, and pure probabilistic graphical models, are necessary to compose a sufficiently general interface layer between artificial intelligence and the algorithms that implement it [9]. Similar to Sigma, models of cognition are implicitly tied to specific inference algorithms in both Alchemy and the SERKET frameworks. Thus these cannot either be considered suitable as a generalized frameworks.

Based on the observation that the layers of Sigma's Cognitive Hourglass model conceptually can be divided into more generalized layers, we propose a generalized Cognitive Hourglass model based on recent advancements within machine learning that makes no such commitments. More specifically, the main contribution of this paper is a framework for developing cognitive architectures for robotics centered around probabilistic programs that

- separates a specific model of cognition from the algorithm that implements it,
- allows the combination of logic and probabilistic models,
- is not tied to specific inference algorithms,
- provides a structure dividing the development of cognitive architectures into layers,
- and embraces the same four desiderata as Sigma.

The presented Generalized Cognitive Hourglass model constitutes a flexible framework for guiding and discussing the future development of cognitive robotics. As such, with this paper we do not intend to construct a new specific cognitive architecture our-self. Our framework should be viewed as a space of systems subsuming Sigma, Alchemy and the SERKET frameworks among others, and our intend with this framework and manuscript is

1. to provide a framework for other researchers to expand upon,
2. to ease the development of cognitive architectures for robotics by encouraging and mitigating cooperation and re-use of existing results,
3. and finally to highlight some of the current state-of-the-art technology available to progress this research field.

In Section A.2 we briefly introduce Sigma's Cognitive Hourglass model in more detail. Based on this, we present our Generalized Cognitive Hourglass model as a framework for developing new cognitive architectures aimed at cognitive robotics in Section A.3. In Section A.5 we give a brief introduction to probabilistic programs since the presented framework is built around them. Explaining the functionality of probabilistic programs with conventional methods can be difficult, therefore in Section A.5.1 we present a graphical representation of probabilistic programs which we call "generative flow graphs". We do so in the hope that it will ease the dissemination of new models of parts of cognition developed within the proposed framework. Being fundamental to achieving functional elegance within the proposed framework, we formally introduce the concept of probabilistic programming idioms in Section A.5.2 and explain how "generative flow graphs" can aid the identification of such idioms. In Section A.6 we discuss the intrinsic problem of performing approximate inference in complex probabilistic programs and present some modern algorithms to tackle this problem for cognitive robotics. As probabilistic programming languages form the foundation of the present framework, we give a brief survey of probabilistic programming languages relevant to the framework in Section A.7. Finally, in Section A.8 we shortly present some preliminary work to support the framework presented.

A.2 Sigma's Cognitive Hourglass Model

Fig. A.1 illustrates how the dimensions of Sigma's Cognitive Hourglass Model relate to the four desiderata. The top layer of the hourglass represents all the knowledge and skills implemented by the cognitive system. This includes high level cognitive capabilities such as reasoning, decision making, and meta cognition, as well as low level cognitive capabilities such as perception, attention, and the formation of knowledge and memory that could potentially be inspired by human cognition. But it also includes artificial cognitive capabilities such as, e.g., the creation of a grid-maps common in robotics. As such, the extend of this layer corresponds to the achievable extend of *Grand Unification* and *Generic Cognition*.

The "cognitive architecture" layer defines central architectural decisions such as the utilization of the cognitive cycle and tri-level control structure for information processing, and the division of memory into a perceptual buffer, working memory and long-term memory. But also defines other architectural concepts such as "functions", "structures", "affect/emotion", "surprise", and "attention". Thereby, the cognitive architecture induces what can be considered an "cognitive programming language" in which all of the knowledge and skills in the top layer can be embodied and learned. As an intermediate layer, cognitive idioms provide design patterns, libraries, and services that ease the implementation of knowledge and skills.

Below the Cognitive architecture and at the waist of the model, they have the graphical architecture constituting a small elegant core of functionality. *Functional elegance* is thereby obtained by compilation of knowledge and skills through a series of layers into a common representation in the graphical architecture. This graphical architecture primarily consists of probabilistic inference over graphical models, more specifically factor graphs, utilizing the sum-product algorithm [18] plus the following extensions:

1. each variable node is allowed to correspond to one or more function variables,
2. special purpose factor nodes,
3. and the possibility of limiting the direction of influence along a link in the graph.

Of these extensions, the two first are merely special-purpose optimizations for the inference algorithm, i.e., a part of the implementation layer in Fig. A.1. According to the authors the third extension has "a less clear status concerning factor graph semantics" [1]. Finally, the graphical architecture is implemented in the programming language LISP. In this model, *sufficient efficiency* is achieved as the cumulative efficiency of all layers. I.e., an efficient implementation in LISP is futile if models of knowledge and skills are inefficient for a given task.

To summarize, the model shown in Fig. A.1 commits to multiple more or less restrictive decisions such as the utilization of factor graphs and the sum-product algorithm at its core, the "cognitive cycle", the tri-level control structure, and LISP as the exclusive implementation language. While these commitments may be suitable for the specific cognitive architecture Sigma mainly targeted human-like intelligence, they would hinder the exploration of new ideas and the utilization of new technologies, making this model less suitable as a general framework.

A.3 Generalized Cognitive Hourglass Model

While Sigma's Cognitive Hourglass model has an advantageous structure with roots in highly appropriate desiderata, it is not suitable as a general framework, due to some exclusive structural commitments. We argue that these structural commitments are mostly artefacts of the limited expressibility of factor graphs and the sum-product algorithm.

Consider, for instance, the "cognitive cycle" dividing processing into an elaboration and adaption phase. The elaboration phase performs inference over the factor graph, while the adaption phase modifies the factor graph before further inference. We argue that this two-phase division of processing is caused by the need for the sum-product algorithm to operate on a static factor graph. This cognitive cycle makes the tri-level control structure necessary to make cognitive branching and recursion possible. Similarly, we argue

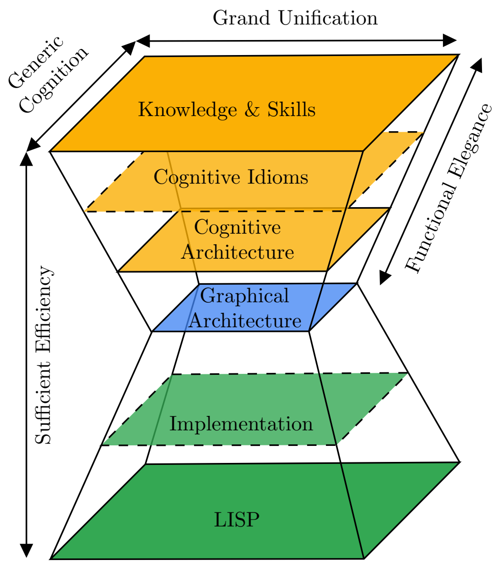


Fig. A.1: Sigma's Cognitive Hourglass Model
Loose re-drawing of figures of the cognitive hourglass model presented in [1]. Layers with dashed borders are not recognized as distinguishable layers by the authors of [1].

that the third extension of the factor graph semantics employed in the cognitive architecture Sigma is nothing more than a simple control flow construct over the information flow in the graphical model and inference algorithm. It is straightforward to imagine how other control flow constructs such as recursion, loops, and conditionals could also be advantageous in modeling cognition.

Basically, we believe that special-purpose implementations of architectural constructs such as the two-phase "cognitive cycle" employed by Sigma and similar cognitive architectures have previously been necessary due to the limitations of the available modeling tools. The flexibility of probabilistic programs provided by the possibility of incorporating I/O operations, loops, branching, and recursion into a probabilistic model should permit representing such constructs as either Probabilistic Programming or Cognitive Idioms instead.

In Fig. A.2 we present our proposal for a more general cognitive hourglass model having probabilistic programs at its waist as the theoretical modeling base. Just like the model in Section A.2, our model is composed of a series of layers that expands away from the waist of the hourglass. On top of the pure probabilistic program, we might be able to recognize program fragments that are sufficiently general to be considered idioms. From these idioms, it might be possible to construct dedicated programming languages for expressing

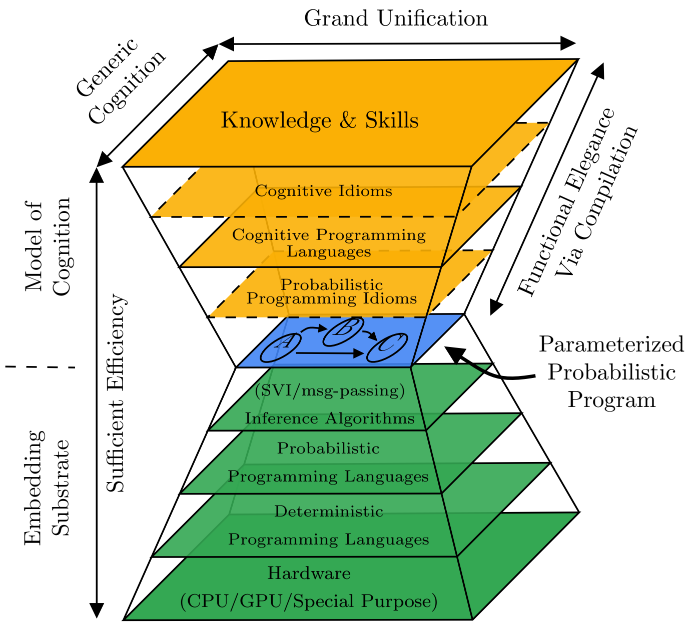

Grand Unification
Generic Cognition
Knowledge & Skills
Cognitive Idioms
Cognitive Programming Languages
Probabilistic Programming Idioms
Functional Elegance Via Compilation
Model of Cognition
Sufficient Efficiency
Embedding Substrate
A
B
C
Parameterized Probabilistic Program
(SVI/msg-passing) Inference Algorithms
Probabilistic Programming Languages
Deterministic Programming Languages
Hardware (CPU/GPU/Special Purpose)

Fig. A.2: The Generalized Cognitive Hourglass Model
Our proposal for a generalized cognitive hourglass model. Dashed borders indicates layers that are not necessarily recognized as distinguishable layers, but could help in the development of the other layers.

cognitive behavior, knowledge, and skills, such as the "Cognitive Language" employed in the cognitive architecture Sigma [1]. In this framework, functional elegance above the probabilistic program is obtained via compilation of knowledge and skills through appropriate cognitive programming languages into probabilistic programs. Below the probabilistic program's different inference algorithms can carry out the necessary inference in the probabilistic program. Different versions of these inference algorithms can potentially be implemented in different probabilistic languages. Both the probabilistic program and probabilistic programming language can be situated in standard deterministic programming languages. Furthermore, one needs not even use the same deterministic programming language for both [19], thereby separating the development of models of cognition from the development of the algorithms that implements them. Finally, the deterministic programming languages allow us to execute a model of cognition on different types of hardware doing the actual computations. When comparing Sigma's Hourglass model to the Generalized Hourglass model the complexity might seem to have increased. However, this is not the case. The Generalized Hourglass model simply highlights some of the components implicit in Sigma's Hourglass model.

We expect that this model is sufficiently general to be considered a framework for research in and development of cognitive robotics, and as stated in Section A.1 the presented model should be considered a space of systems subsuming others. Our model subsumes Sigma, which limits the Probabilistic programs at the waist of our model to factor graphs, and limits inference to the sum-product algorithm. As another example, consider the SERKET frameworks. In both frameworks, exact message-passing is used to perform inference on probabilistic graphical models with discrete and finite variables, and otherwise, sampling importance resampling is used. Both of these frameworks can also be considered special cases of our framework, with "modules" and their connections somehow resembling what we have chosen to call "probabilistic programming idioms". The "modules" in SERKET and Neuro-SERKET are supposed to be fully defined and self-contained. In contrast, our definition of "probabilistic programming idioms" allows for nesting and e.g. class definitions with abstract methods as we will exemplify in Section A.8. As a third example, Alchemy may also be considered one instance of our framework, limiting the probabilistic programs at the waist to Markov logic, and utilizing a combination of Markov chain Monte Carlo and lifted belief propagation for inference [9]. In fact, as probabilistic programs can be considered an extension of deterministic programs, it should even be possible to situate both emergent, symbolic, and hybrid approaches to cognitive architectures in this framework, thereby covering the full taxonomy considered in [4]. In our framework constructs such as the cognitive cycle and tri-level control structure could potentially be expressed as probabilistic programming idioms rather than special-purpose architectural implementations. Similarly, incorporation of results from other research areas such as deep learning is only limited to the extent that a given probabilistic programming language and corresponding inference algorithms can incorporate essential tools used in these research areas, i.e., automatic differentiation for deep learning. Furthermore, this framework gives a satisfying view on the foundational hypothesis in artificial intelligence about substrate independence [10], by cleanly separating the model of cognition, i.e. the probabilistic program and everything above it, from the organic or inorganic substrate that it exists on, i.e., everything below the probabilistic program.

While the above might sound promising, the choice of probabilistic programs as a focal point also has important ramifications. In general, we cannot guarantee the existence of an analytic solution for all models, and even if a solution exists it might be computationally intractable [19]. Therefore, we have to endure approximate solutions. Though this might sound restrictive, this is also the case for most other complex real-world problems. In fact, it can be considered a form of bounded rationality consistent with the concept of "satisficing" stating that an organism confronted with multiple goals does not have the senses nor the wits to infer an "optimal" or perfect solution, and thus will settle for the first solution permitting satisfaction at some specified level of all of its needs [20]. The second important ramification is that the model with its roots in probabilistic and deterministic programming languages is only applicable to the extent to which the hypothesis that artificial cognition can be grounded in such programming languages is valid. However, this is currently a widely accepted hypothesis.

It is important to stress that the layers of the proposed framework are not independent. On the contrary, as the technological possibilities and community knowledge evolves, changes in one layer might open new possibilities in the layers above. Similarly, the need for new features in one layer might guide the research directions and development of the layers below. However, this structure is exactly what would allow further discussions and development in cognitive robotics to evolve at different levels of abstractions, and benefit from other research fields related to the layers below probabilistic programs. In the layers above probabilistic programs, the development and identification of both probabilistic programming idioms, cognitive programming languages, and cognitive idioms mitigate cooperation and re-use of existing results. The framework thus minimizes the burden of developing new cognitive architectures by allowing researchers to focus their energy on specific layers, or parts thereof, in the hourglass model rather than dealing with all the details of a cognitive architecture. The extent to which the burden of development is reduced thus depends upon the technology available in each of the layers of the hourglass.

A.4 Preliminaries

In this paper, we do not distinguish between probability density functions and probability mass functions, and jointly denote them as probability functions. The symbol $\int$ is used to denote both integrals and summations depending on the context. In general, we use z to denote latent random variables, x to denote observed random variables, $p(...)$ to denote "true" probability functions, $q(...)$ to denote approximations to "true" probability functions, θ to denote parameters of "true" probability functions, $p(...)$, and ϕ to denote parameters of approximations to "true" probability functions, $q(...)$. When a probability function directly depends on a parameter we write the parameter in a subscript before the parenteses, e.g., $p_\theta(...)$ and $q_\phi(...)$. We use line over a value, parameter, or random variable to denote that it is equal to a specific value, e.g., $\overline{z} = 1,432$. We use a breve over a parameter or random variable to denote that it should be considered a fixed parameter or random variable within that equation, e.g., $\breve{\theta}$ or $\breve{z}$. For parameters this means that they attain a specific value, $\overline{\theta}$, i.e., $\breve{\theta}$ means that $\theta = \overline{\theta}$. For random variables it means that the probability functions that this variable is associated with is considered fixed within a given equation. We use capital letters to denote sets, e.g., $A = \{1, ..., \overline{n}\}$. We use a superscript with curly brackets to denote indexes. E.g. $z^{\{i\}}$ would denote the i'th latent random variable. Similarly, we use a superscript with curly brackets and two numbers separated by a semicolon to denote a set of indexes values, i.e., $z^{\{1;\overline{n}\}} = z^{\{A\}} = \left\{z^{\{1\}}, ..., z^{\{\overline{n}\}}\right\}$. We use a backslash, $\setminus$, after a set followed by a value, random variable, or parameter to denote the exclusion of that value, random variable, or parameter from that set, i.e., $z^{\{A\}} \setminus z^{\{\overline{n}\}} = \left\{z^{\{1\}}, ..., z^{\{\overline{n}-1\}}\right\}$. We use capital C to denote a collection of latent random variables, observed random variables, and parameters. Furthermore, we will specify such a collection by enclosing variables and parameters with curly brackets around and with a semi-colon separating latent random variables, observed random variables, and parameters in that order, e.g., $C = \{Z; X; \Theta\}$. We will use Pa, Ch, An, and De as abbreviations for parent, child, ancestors, and descendants, respectively, and use, e.g., $\text{Pa}\Theta(C)$ to denote the set of parameters parent to the collection C, and $\text{Ch}X(Z)$ to denote the set of observed variables that are children of the latent random variable Z.

A.5 Probabilistic Programs

At the heart of our framework, we have chosen to place probabilistic programs. One definition of probabilistic programs is as follows:

"Probabilistic programs are usual functional or imperative programs with two added constructs: (1) the ability to draw values at random from distributions, and (2) the ability to condition values of variables in a program via observations." [21]

With these two constructs, any functional or imperative program can be turned into a simultaneous representation of a joint distribution, $p_\Theta(Z, X)$, and conditional distribution, $p_\Theta(X|Z)$, where X represent the conditioned/observed random variables, Z the unconditioned/latent random variables, and Θ represents other parameters in the program that are not given a probabilistic treatment. Thereby allowing us to integrate classical control constructs familiar to any programmer such as if/else statement, loops, and recursions into probabilistic models. As such probabilistic programs can express exactly the same functionality as any deterministic programs can and even more. These two constructs are usually provided as extensions to a given programming language through special *sample* and *observe* functions or keywords [19]. Thus it would be natural to represent such probabilistic programs by pseudocode. However, based on experiences it can be hard to follow the generative

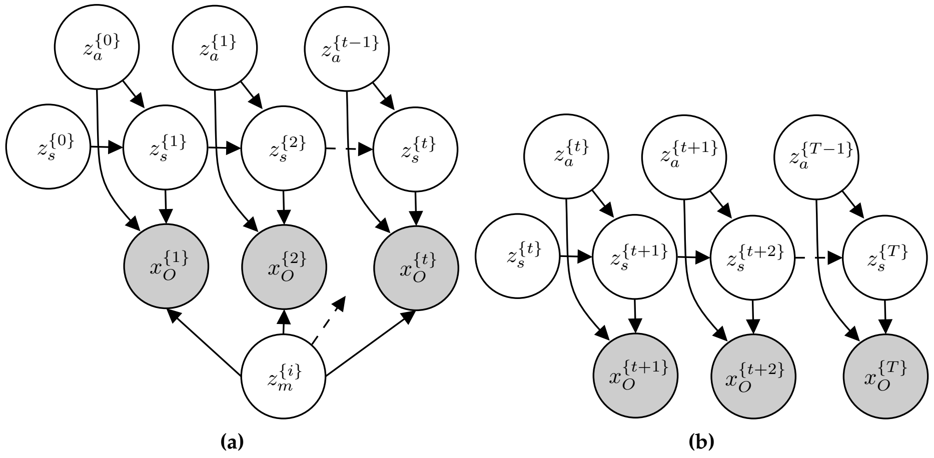


Fig. A.3: Graphical Models With Same Structure
Examples of two directed graphical models developed in different research areas. (a) Graph idiom for the classical simultaneous localization and mapping (SLAM) problem [23]. $z_s^{\{t\}}$ is the state at time t, $z_a^{\{t\}}$ is the action at time t, $z_{map}^{\{i\}}$ is the i'th pixel in a grid map, and $x_p^{\{t\}}$ is the perceived information at time t. (b) Graph idiom for a Markov Decision Process [24]. $z_s^{\{t\}}$ is the state at time t, $z_a^{\{t\}}$ is the action at time t, and $x_O^{\{t\}}$ is a "observed" optimality variable at time t.

flow of random variables in such pseudo-code. Alternatively, such generative flows have classically been represented by directed graphical models [22]. Unfortunately, we have also found that the semantics of classical directed graphical models neither provide an appropriate presentation.

A.5.1 Generative Flow Graphs

We have found that combining the semantics of classical directed graphical models with the semantics of flowcharts into a hybrid representation is a good visual representation. Directed graphical models represent the conditional dependency structure of a model and flowcharts represent the steps in an algorithm or workflow. The hybrid representation illustrates the order in which samples of random variables in a probabilistic program are generated and how these samples influence the distributions used to generate other samples. For this reason, we denote this hybrid representation by the name *Generative Flow Graph*.

To exemplify the utility of the *Generative Flow Graph* representation consider the graphical model for a classical Markov Decision Process and the simultaneous localization and mapping (SLAM) problem depicted in Fig. A.3. With the classical semantics of directed graphical models, it is often the case that size limitations of figures coerce authors to remove some variables from the figure and represent them indirectly by, e.g., dashed arrows as the case in both Fig. A.3a and Fig. A.3b. Similarly, the classical semantics of directed graphical models does not represent the influence from other parameters or variables that are not given a probabilistic treatment, even though such variables and parameters might have equal importance for a model. This is especially true if they are not fixed and have to be learned, e.g., if one wants to incorporate artificial neural networks into a model. The classical semantics of directed graphical models also cannot represent dependency structures depending on conditionals giving the illusion that a variable always depends on all of its possible parents, and that all variables in the graph are relevant in all situations. Furthermore, while the semantics of directed graphical models allows us to represent the structure of the joint distribution, $p(Z, X)$, its ability to explicitly express the structure of the posterior distribution, $p(Z|X)$, is limited. Finally, there is no standardized ways of representing a fragment of a graphical model, which hinders discussions at different levels of abstraction. Probabilistic programs easily allow us to incorporate the above in our models and thus a more appropriate representation is needed. The semantics of generative flow graphs shown in Table A.1 in Appendix A.1 alleviate these problems. Utilizing these semantics we can redraw the directed graphical model in Fig. A.3a in multiple ways with different levels of information as in Fig. A.4. Notice, that the choice of node collections is not unique.

One advantage of the semantics of directed graphical models is that for graphs with no cycles [22] such models represents a specific factorization of the joint probability of all the random variables in the model of the form:

$$p(x^{\{1;\overline{n}\}}, z^{\{1;\overline{m}\}}) = \prod_{n=1}^{\overline{n}} p(x^{\{n\}}|\mathrm{PaZ}(x^{\{n\}})) \prod_{m=1}^{\overline{m}} p(z^{\{m\}}|PaZ(z^{\{m\}})) \tag{A.1}$$

where $x^{\{n\}}$ and $z^{\{m\}}$ are the n'th observed and the m'th latent random variable in the model, respectively. This is in principle also true for the generative flow graph representation if it neither contains any cycles, just with the additional explicit representation of dependency on parameters. For generative flow graphs, we can similarly to Eq. (A.1) write up a factorization by including a factor of the form

$$p_{\mathrm{Pa}\theta(z^{\{m\}})}(z^{\{m\}}|\mathrm{PaZ}(z^{\{m\}}))$$

for each latent random variable node, $z^{\{m\}}$, in the graph, and a factor of the form

$$p_{\mathrm{Pa}\theta(x^{\{n\}})}(x^{\{n\}}|\mathrm{PaZ}(x^{\{n\}}))$$

for each observed random variable node, $x^{\{n\}}$, in the graph, and finally a

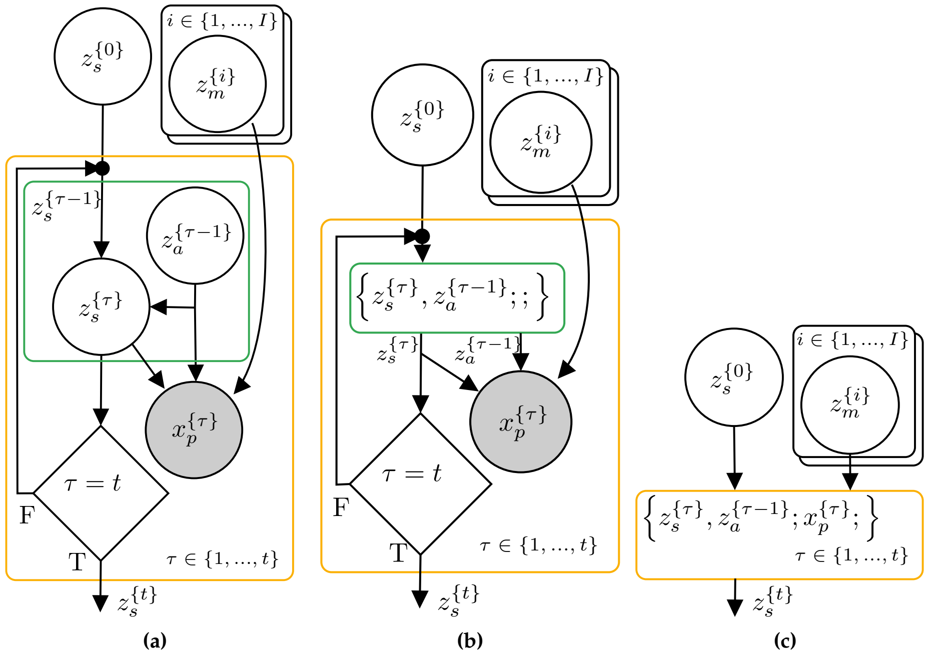


Fig. A.4: Generative Flow Graphs for SLAM
Three semantically equivalent generative flow graphs with different levels of abstractions corresponding to the directed graphical model in Fig. A.3a.

factor of the form

$$p_{\Theta,\mathrm{Pa}\Theta(C^{\{k\}})}(Z, X|\mathrm{Pa}Z(C^{\{k\}})) \tag{A.2}$$

for each node collection, $C^{\{k\}} = \{Z; X; \Theta\}$. If a parent node of y is a node collection $\{Z; X; \Theta\}$ then $\mathrm{Pa}Z(y) = Z$ and $\mathrm{Pa}\theta(y) = \Theta$ unless a subset of the variables or parameters in the node collection is explicitly specified next to the parent link. If the internal structure of a node collection is known from somewhere else, the factor in Eq. (A.2) can of course be replaced by the corresponding factorization. The catch, however, is that a probabilistic program, and thus also generative flow graphs, potentially can denote models with an unbounded number of random variables and parameters making it impossible to write up the full factorization explicitly. On the other hand, this just emphasizes the need for alternative ways of representing probabilistic programs other than pseudo-code.

Besides the possibility of expressing a factorization of the joint prior distribution, the detached link allows us to express additional structure for the posterior distribution, $p(z|x)$. Consider the two generative flow graphs in Fig. A.5. By applying standard manipulations we can obtain the factoriza-

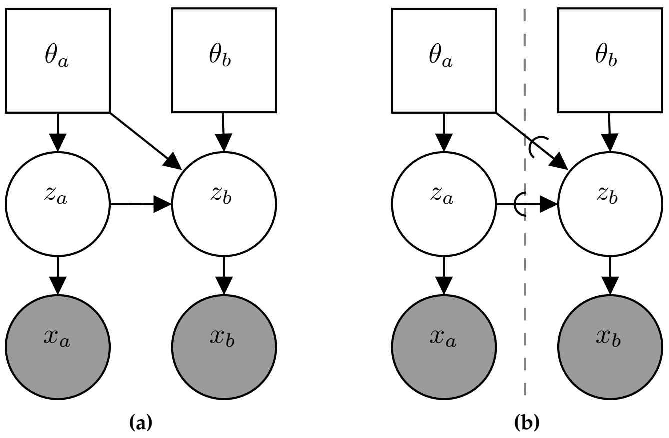


Fig. A.5: Generative Flow Graphs with/-out Detached link
Two generative flow graphs representations of a simple model with two parameters, two latent variables, and two observed variables. In both graphs θ_a and z_a are needed to generate z_b, however, in (b) we have explicitly constrained the inference of z_b to not influence the learning of θ_a and inference of z_a. Thus, the evidence provided by x_b neither is allowed to have an influence on θ_a and z_a. Thereby, the model represented by the nodes on the left-hand side of the dashed line in (b) can be seen as an independent problem.

tion in Eq. (A.3) for the graph in Fig. A.5a.

$$\begin{aligned} p_{\theta_a,\theta_b}(z_a, z_b|x_a, x_b) &= p_{\theta_a,\theta_b}(z_b|z_a, x_a, x_b)p_{\theta_a,\theta_b}(z_a|x_a, x_b) \\ &= p_{\theta_a,\theta_b}(z_b|z_a, x_b)p_{\theta_a,\theta_b}(z_a|x_a, x_b) \end{aligned} \tag{A.3}$$

wheres from the definition of the detached link we can write the factorization in Eq. (A.4) for the graph in Fig. A.5b.

$$\begin{aligned} p_{\theta_a,\theta_b}(z_a, z_b|x_a, x_b) &= p_{\breve{\theta}_a,\theta_b}(z_b|\check{z}_a, \check{x}_a, x_b)p_{\theta_a}(z_a|x_a) \\ &= p_{\breve{\theta}_a,\theta_b}(z_b|\check{z}_a, x_b)p_{\theta_a}(z_a|x_a) \end{aligned} \tag{A.4}$$

The main difference between these two factorizations is the distribution over the latent variable z_a. In Eq. (A.3) the distribution over the latent variable z_a depends on the evidence provided by both observations x_a and x_b, and is influenced by both parameters θ_a and θ_b. In Eq. (A.4) the distribution over z_a depends only on the evidence provided by the observations x_a, and is only influenced by the parameter θ_a. As such the inference problem of obtaining the posterior distribution over z_a is independent of the inference problem of obtaining the posterior distribution over z_b, but not conversely. In general, for model consisting of $\bar{a}$ node collections, $C^{(a)} = \left\{Z^{(a)}; X^{(a)}; \Theta^{(a)}\right\}$, connected only by detached links we can write the factorization of the posterior as

$$p_{\Theta}(Z|X) = \prod_{a=1}^{\bar{a}} p_{\Theta^{\{a\}}, Pa\breve{\Theta}\left(C^{\{a\}}\right)} \left(Z^{\{a\}} | Pa\breve{Z}\left(C^{\{a\}}\right), X^{\{a\}}\right) \tag{A.5}$$

where the breves are used to emphasize that the variables and parameters are related through a detached link. The possible benefit of being able to express such structure will become clear in Section A.6.

Another added benefit of the generative flow graph representation is to express models by different levels of abstraction. As an example consider the three different factorization of the simultaneous localisation and mapping problem given in Eq. (A.6), Eq. (A.7), and Eq. (A.8).

$$\begin{aligned}
&p\left(z_s^{\{0;t\}}, z_a^{\{0;t-1\}}, x_p^{\{1;t\}}, z_m^{\{0;I\}}\right) \\
&= p\left(z_s^{\{0;t\}}, z_a^{\{0;t-1\}}, x_p^{\{1;t\}} | z_s^{\{0\}}, z_m^{\{0;I\}}\right) p\left(z_s^{\{0\}}\right) \prod_{i=1}^{I} p\left(z_m^{\{i\}}\right) && \text{(A.6)} \\
&= p\left(z_s^{\{0\}}\right) \prod_{i=1}^{I} p\left(z_m^{\{i\}}\right) \prod_{\tau=1}^{t} \left(\begin{array}{c} p\left(x_p^{\{\tau\}} | z_s^{\{\tau\}}, z_a^{\{\tau-1\}}, z_m^{\{0;I\}}\right) \\ \cdot p\left(z_s^{\{\tau\}}, z_a^{\{\tau-1\}} | z_s^{\{\tau-1\}}\right) \end{array} \right) && \text{(A.7)} \\
&= p\left(z_s^{\{0\}}\right) \prod_{i=1}^{I} p\left(z_m^{\{i\}}\right) \prod_{\tau=1}^{t} \left(\begin{array}{c} p\left(x_p^{\{\tau\}} | z_s^{\{\tau\}}, z_a^{\{\tau-1\}}, z_m^{\{0;I\}}\right) \\ \cdot p\left(z_s^{\{\tau\}} | z_a^{\{\tau-1\}}, z_s^{\{\tau-1\}}\right) p\left(z_a^{\{\tau-1\}}\right) \end{array} \right) && \text{(A.8)}
\end{aligned}$$

The generative flow graphs in Fig. A.4a, Fig. A.4b, and Fig. A.4c corresponds directly to the factorization in Eq. (A.8), Eq. (A.7), and Eq. (A.6), respectively. Thereby, they represents different levels of abstractions for the same model. As such generative flow graphs simply yield better expressibility over their directed graphical model counterparts.

A.5.2 Probabilistic Programming Idioms

We have already discussed how probabilistic programming idioms can be seen as a means to achieve functional elegance. In this section, we describe how such idioms can be discovered by inspecting generative flow graphs. We define probabilistic programming idioms as follows:

"Probabilistic programming idioms are reusable code fragments of probabilistic programs sharing an equivalent semantic role in their enclosing probabilistic programs."

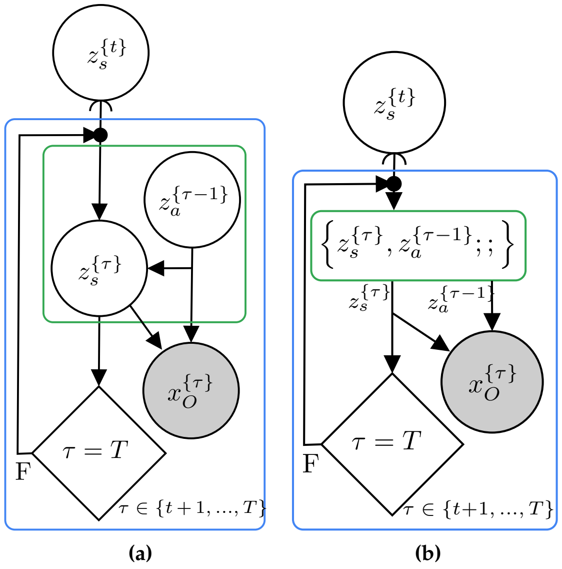


Fig. A.6: Generative Flow Graphs for a Markov Decision Process
Two semantically equivalent generative flow graphs corresponding to the directed graphical model in Fig. A.3b.

To identify such probabilistic programming idioms, we can look for node collections containing the same nodes and with the same internal structure in at least two different probabilistic programs. Consider for example the node collection $\left\{z_s^{\{\tau\}}, z_a^{\{\tau-1\}};;\right\}$ highlighted with a green border in the generative flow graph for both the simultaneous localization and mapping problem and Markov Decision Process depicted in Fig. A.4 and Fig. A.6, respectively. From Fig. A.4a and Fig. A.6a it is clear that the internal structure of this node collection is identical in both graphs, and that it represents the factorization

$$p\left(z_s^{\{\tau\}} | z_s^{\{\tau-1\}}, z_a^{\{\tau-1\}}\right) p\left(z_a^{\{\tau-1\}}\right).$$

Assuming that the distributions $p\left(z_s^{\{\tau\}} | z_s^{\{\tau-1\}}, z_a^{\{\tau-1\}}\right)$ and $p\left(z_a^{\{\tau-1\}}\right)$ are the same in both models, we could possible create a probabilistic program for this node collection once, and then reuse it in both models. This probabilistic program should then take a sample $z_s^{\{\tau-1\}}$ as input. From this input the program could sample both $z_s^{\{\tau\}}$ and $z_a^{\{\tau-1\}}$ from "hard-coded" distributions $p\left(z_s^{\{\tau\}} | z_s^{\{\tau-1\}}, z_a^{\{\tau-1\}}\right)$ and $p\left(z_a^{\{\tau-1\}}\right)$ using the sample function or keyword of the probabilistic programming language. Finally, the program should return both of these samples. While this approach might work perfectly for some applications, the two distributions $p\left(z_s^{\{\tau-1\}} | z_s^{\{\tau-1\}}, z_a^{\{\tau-1\}}\right)$

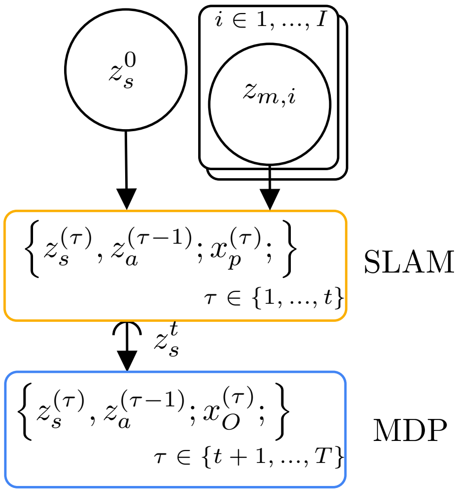

z_s^0
$i \in 1, \dots, I$
$z_{m,i}$
$\left\{ z_s^{(\tau)}, z_a^{(\tau-1)}; x_p^{(\tau)}; \right\}$
$\tau \in \{1, \dots, t\}$
SLAM
z_s^t
$\left\{ z_s^{(\tau)}, z_a^{(\tau-1)}; x_O^{(\tau)}; \right\}$
$\tau \in \{t+1, \dots, T\}$
MDP

Fig. A.7: Composition of Generative Flow Graphs
A combination of the generative flow graphs for the simultaneous localization and mapping problem and the Markov Decision Process shown in Fig. A.4 and Fig. A.6, respectively, could potentially constitute an end-to-end navigation behavior for a mobile robot.

and $p\left(z_a^{\{\tau-1\}}\right)$ are usually application specific, limiting the usability for an idiom in which they are "hard-coded". A far more general approach would be to allow the probabilistic program to instead take the two distributions as input or having these distributions as free variables, allowing us to re-use the code fragment even for problems where these distributions are not necessarily the same. Rather than fully defining a model of a part of cognition, such a probabilistic program would constitute a template method for the generative flow of that part of cognition. Specific utilization of the model could then be done via a function closure specifying the free distributions. While the benefits of the above example arguably might be limited since the internal structure of the node collection is relatively simple, it is not hard to imagine more complex structures. Consider for instance the node collection highlighted with a blue border in Fig. A.6. By constructing an appropriate probabilistic program for this node collection we have defined a probabilistic programming idiom constituting the foundation for optimal control and reinforcement learning.

When we have developed such probabilistic programming idioms it empowers us to mix and match them to construct higher-level intelligence without worrying about all details of the underlying models. E.g. Fig. A.7 implies that the output of a specific model for the simultaneous localization and mapping problem is used as the input to a Markov decision process, but leaves out details about their internal structures.

A.6 Inference Algorithms

As stated in Section A.5 a probabilistic program is a simultaneous representation of a joint distribution, $p_\Theta(Z, X)$, and a conditional distribution, $p_\Theta(X|Z)$. Having defined a model as such distributions we are usually interested in answering queries about the unconditioned/latent random variables, Z, given information about the conditioned/observed random variables, $X = \overline{X}$. In the combined navigation problem illustrated in Fig. A.7 we are interested in determining which action to take, $z_a^{\{\tau\}}$, given prior perceived information, $x_p^{\{\tau\}}$ for $\tau \in 1, ..., t$, and future optimality variables, $x_O^{\{\tau\}}$ for $\tau \in t+1, ..., T$. Often queries of interest are statistics such as the posterior mean and variance of specific random variables, or the posterior probability of a random variable being within a given set. Still, it could also simply be to sample from the posterior, $p_\Theta\left(Z|X = \overline{X}\right)$. All of these queries are somehow related to the posterior distribution given by

$$\begin{aligned} p_\Theta\left(Z|X = \overline{X}\right) &= \frac{p_\Theta\left(X = \overline{X}, Z\right)}{p_\Theta\left(X = \overline{X}\right)} \\ &= \frac{p_\Theta\left(X = \overline{X}, Z\right)}{\int p_\Theta\left(X = \overline{X}, Z\right) dZ}. \end{aligned} \tag{A.9}$$

The marginalization by the integral in the denominator of Eq. (A.9) in general does not have an analytical solution or is intractable to compute in most realistic problems and approximate inference is therefore necessary [25]. Through time, an abundance of algorithms has been developed to find an approximation to the posterior in specific problems. Unfortunately, many of these algorithms cannot be applied to general probabilistic programs mainly due to the possible unbounded number of random variables [19]. Possible applicable inference algorithms can roughly be divided into Monte Carlo based algorithms such as Sequential Monte Carlo, Metropolis-Hastings, and Hamiltonian Monte Carlo, and the optimization based Variational Inference algorithms such as Stochastic Variational inference. As the size and complexity of models of cognition increase, the computational efficiency of inference algorithms becomes a paramount necessity to achieve sufficient efficiency of the framework presented in Section A.3. While Monte Carlo methods often converge to the true posterior in the limit, convergence can be slow. Conversely, Variational Inference algorithms are often faster even though they can suffer from simplified posterior approximations [25]. Also, as Variational Inference methods are based on optimization they provide a natural synergy with data-driven discriminative techniques such as deep learning. By accepting that robots are not necessarily supposed to behave optimally, but rather should behave as agents with bounded rationality, the above characteristic makes Variational Inference algorithms an especially interesting choice for cognitive robotics. Therefore, Section A.6.1 is devoted to giving the reader an introduction to the overall concept of Variational Inference. Section A.6.2 and A.6.3 present two specific solution approaches commonly used in variational inference, namely Message-passing algorithms and stochastic variational inference, respectively. Both approaches have their weaknesses. Therefore, in Section A.6.4 we outline a way of combining these two approaches to overcome their weaknesses. The idea of combining Message-passing with stochastic variational inference, we have presented before [26], however, here we generalizes the idea to generative flow graphs.

A.6.1 Variational inference

Variational inference is an optimization based approach to approximate one distribution, $p\left(Z\right)$, by another simpler distribution, $q\left(Z\right)$. $q\left(Z\right)$ is usually called the variational distribution. In general, variational inference is not only used to approximate conditional distribution, $p\left(Z|X=\overline{X}\right)$, as in Eq. (A.9). However, with the presented framework in mind we will limit our presentation to this case, and focus on variational inference problems on the form

$$q^{*}\left(Z\right)=\arg\min_{q(Z)\in Q} D\left(p_{\Theta}\left(Z|X=\overline{X}\right)||q\left(Z\right)\right) \tag{A.10}$$

where D is a measure of the similarity between p and q often called a divergence measure, and Q is the family of variational distributions from which the approximation should be found. The notation $D(p||q)$ denotes a divergence measure and that the order of the arguments, p and q, matters. The choice the family of variational distributions, Q, is a compromise between computational efficiency and precise an approximation one wants. Furthermore, Q should be chosen such that we can easily answer given queries. It is important to stress that any variational inference method is more or less biased via the choice of the family of variational distributions, Q. As a consequence we cannot view the original model in isolation, and have to consider the variational distribution, $q\left(Z\right)$, as an implicit part of the cognitive model. Besides the family of variational distributions, the choice of the divergence measure, D, can substantially impact the properties of the approximation. However, empirical results suggest that for the family of α-divergences, subsuming the commonly used Kullback–Leibler divergence, all choices will give similar results as long as the approximating family, Q, is a good fit to the true posterior distribution [27].

A.6.2 Message-Passing

Message-passing algorithms solves a possible complicated variational inference problem as defined by Eq. (A.10) by it down into a series of more

```
1: Initialize $q^{\{a\}*}(Z)$ for all $a \in A$
2: **repeat**
3:     Pick a factor $a \in A$
4:     Solve Eq. (A.15) to find $q^{\{a\}*}(Z)$
5: **until** $q^{\{a\}*}(Z)$ converges for all $a \in A$
```

Algorithm A.1: Generic Message-Passing Algorithm
Pseudocode for the Generic Message-Passing Algorithm. The loop in line 2 can potentially be run in parallel and in a distributed fashion.

tractable sub-problems [27]. The methods are known as message-passing algorithms due to the way that the solution to one sub-problem is distributed to the other sub-problems. Message-passing algorithms assumes that the model of a problem, $p(Z|X)$, can be factorized into a product of probability distributions

$$p(Z|X) = \prod_{a \in A} p^{\{a\}}(Z|X). \tag{A.11}$$

This factorization need not be unique and each factor, $p^{\{a\}}(Z|X)$, can depend on any number of the variables of $p(Z|X)$. The variational distribution, $q\,(Z)$, should furthermore be chosen such that it factorizes into a similar form

$$q\,(Z) = \prod_{a \in A} q^{\{a\}}(Z). \tag{A.12}$$

With these assumptions, define the product of all other than the a'th factor of $q\,(Z)$ and $p(Z|X)$, respectively as

$$q^{\backslash a}\,(Z) = \prod_{b \in A \backslash a} q^{\{b\}}(Z), \tag{A.13}$$

$$p^{\backslash a}\,(Z|X) = \prod_{b \in A \backslash a} p^{\{b\}}(Z|X). \tag{A.14}$$

With these definitions it is possible to rewrite the problem in Eq. (A.10) into a series of approximate sub-problems on the form

$$q^{\{a\}*}(Z) \approx \arg \min_{q^{\{a\}} \in Q^{\{a\}}} D\left[p^{\{a\}}(Z|X) q^{\backslash a}\,(Z) || q^{\{a\}}(Z) q^{\backslash a}\,(Z)\right] \tag{A.15}$$

where $q^{\backslash a}\,(Z)$ is assumed to be a good approximation and thus is kept fixed. If the factor families, $Q^{\{a\}}$, from which $q^{\{a\}}$ can be chosen, have been chosen sensible, the problem in Eq. (A.15) can be more tractable than the original problem, and an approximate solution to the original problem can then be obtain by iterating over these coupled sub-problems as shown in Algorithm A.1. In principle, we can even use different divergence measures for each sub-problem to do mismatched message-passing, which could make some of the sub-problems easier to solve as described in [27]

In general, the approach is not guaranteed to converge, and Eq. (A.15) might still be a hard problem to solve. In the past, this has limited the approach to relatively simple problems such as fully discrete or Gaussian problems for which Eq. (A.15) can be solved analytically [27]. Therefore, the true power of the method is the principle way in which it allows for solving problems in a distributed and parallel fashion, which can be a huge benefit for large models. Furthermore, if the sub-problems are sparsely connected, meaning that sub-problems does not depend on the solution to all of the other sub-problems, the amount of communication needed can be significantly reduced.

A.6.3 Stochastic Variational inference

The approach taken by Stochastic Variational inference (SVI) is to reformulate a variational inference problem, e.g., Eq. (A.10) or Eq. (A.15), to a dual maximization problem with an objective, L, that that can be solved with stochastic optimization [28]. Stochastic Variational inference assumes that the variational distribution, q, is parameterized by some parameters, Φ. To obtain the dual problem and the objective function, L, of the resulting maximization problem of course the steps and assumptions needed depends on whether we have chosen the use the Kullback–Leibler divergence [28], [29], [30] or the α-divergences [31]. Nevertheless, the resulting problem ends up being on the form

$$\Phi^* = \arg\max_{\Phi} \quad \underbrace{L\left(p_\Theta\left(Z, X = \overline{X}\right), q_\Phi\left(Z\right)\right)}_{E_{Z\sim q_\Phi(Z)}[l(Z,\Theta,\Phi)]}. \tag{A.16}$$

This dual objective function, L, does not depend on the posterior, $p_\Theta\left(Z|X = \overline{X}\right)$, but only the variational distribution, $q_\Phi\left(Z\right)$ and the unconditional distribution $p_\Theta\left(Z, X = \overline{X}\right)$ making the problem much easier to work with. Besides being dual to Eq. (A.10) it turns out that for the family of alpha-divergences with $\alpha > 0$, L is also an lower bound on the log evidence, $log\left(p_\Theta\left(Z\right)\right)$ [31]. Since the log evidence is a measure of how well a model fits the data, we can instead consider the optimization problem [32]

$$\Theta^*, \Phi^* = \arg\max_{\Theta,\Phi} \quad \underbrace{L\left(p_\Theta\left(Z, X = \overline{X}\right), q_\Phi\left(Z\right)\right)}_{E_{Z\sim q_\Phi(Z)}[l(Z,\Theta,\Phi)]}. \tag{A.17}$$

that allows us to simultaneously fit the posterior approximation, q_Φ, and model parameters, Θ, to the data, $\overline{X}$. An unbiased estimate of the gradient, $\overline{\nabla_W L}$, of this dual objective L where $W = \{\Theta, \Phi\}$, can be obtained by utilizing the REINFORCE-gradient [33] or the reparameterization trick [32], [34], [35].

The objective can then iteratively be optimized by stochastic gradient ascent via the the update equation

$$W^{\{l\}} = W^{\{l-1\}} + \rho^{\{l-1\}} \overline{\nabla_W L}^{\{l\}} \left(W^{\{l-1\}} \right) \tag{A.18}$$

where superscript $\{l\}$ is used to denote the l'th iteration. Stochastic gradient ascent converges to a maximum of the objective function L if the sequence of learning rates, $\rho^{\{l-1\}}$, follows the Robbins-Monro conditions given by

$$\sum_{l=1}^{\infty} \rho^{\{l\}} = \infty, \qquad \sum_{l=1}^{\infty} \left(\rho^{\{l\}} \right)^2 < \infty. \tag{A.19}$$

Since Eq. (A.17) is dual to the original minimization problem, this maximum also provides a solution to the original problem. Although Robbins-Monro conditions are satisfied it is often necessary to apply variance reduction methods to obtain unbiased gradient estimators with sufficiently low variance. Fortunately, reduction methods can often be applied automatically by probabilistic programming libraries/languages such as Pyro [36]. One benefit of solving variational inference problems with stochastic optimization is that noisy gradient estimates are often relatively cheap to compute due to, e.g., subsampling of data. Another benefit is that the use of noisy gradient estimates can cause algorithms to escape shallow local optima of complex objective functions [28]. The downside of Stochastic variational inference is that it is inherently serial and that it requires the parameters to fit in the memory of a single processor [37]. This could potentially be a problem for cognitive robotics where large models with lots of variables and parameters presumably are necessary to obtain a high level of intelligence, and where queries have to be answered within different time scales. I.e. signals to motors have to be updated frequently while high-level decisions can be allowed to take a longer time.

A.6.4 Stochastic Message-Passing

To summarize the previous sections Message-passing algorithms exploits the dependency structure of a given variational inference problem to decompose the overall problem into a series of simpler variational inference sub-problems, that can be solved in a distributed fashion [27]. Message-passing algorithms do not give specific directions on how to solve these sub-problems, and thus classically required tedious analytical derivations, that effectively limited the usability of the method. On the other hand, modern Stochastic variational inference methods directly solve such variational inference problems utilizing stochastic optimization while simultaneously learning model parameters. By fusion of these two approaches, we could potentially overcome the serial nature of Stochastic variational inference to solve large-scale complex problems in a parallel and distributed fashion. However, to do so we need to find an appropriate factorization of a given problem. Again we can make use of the semantics of generative flow graphs. Assuming that we can divide all the nodes of a given generative flow graph into a set, $C^{\{A\}}$, of node collections $C^{\{a\}} = \left\{ Z^{\{a\}}; X^{\{a\}}; \Theta^{\{a\}} \right\}$ and a set of "global" observed variable nodes, X_G, having more than one node collection as parent, we can write the posterior factorization

$$\begin{aligned} p_\Theta(Z|X) &= p_\Theta(Z^{\{A\}}|X^{\{A\}}, X_G) \\ &= \frac{1}{p(X_G|X^{\{A\}})} p_\Theta(Z^{\{A\}}, X_G|X^{\{A\}}) \\ &= \frac{p(X_G|Z^{\{A\}}, X^{\{A\}})}{p_\Theta(X_G|X^{\{A\}})} p_\Theta(Z^{\{A\}}|X^{\{A\}}) \\ &= \frac{p(X_G|Z^{\{A\}})}{p_\Theta(X_G|X^{\{A\}})} p_\Theta(Z^{\{A\}}|X^{\{A\}}) \qquad \text{(A.20)} \\ &= \frac{p(X_G|Z^{\{A\}})}{p_\Theta(X_G|X^{\{A\}})} \prod_{a\in A} p_{\Theta^{\{a\}},\mathrm{Pa}\check{\Theta}\left(C^{\{a\}}\right)} \left(Z^{\{a\}}|\mathrm{Pa}\check{Z}\left(C^{\{a\}}\right), X^{\{a\}} \right) \\ & \qquad \text{(A.21)} \end{aligned}$$

where Eq. (A.20) follows from conditional independence between X_G and $X^{\{A\}}$ given $Z^{\{A\}}$, and Eq. (A.21) follows from Eq. (A.5). Following the procedure of message-passing we choose a variational distribution that factorizes as

$$q_\Phi(Z) = \prod_{a\in A} q_{\Phi^{\{a\}}} \left(Z^{\{a\}} \right) \qquad \text{(A.22)}$$

Notice, that in Eq. (A.22) we have exactly one factor for each node collection and that this factor only contains the latent variables of that node collection. This is unlike Eq. (A.12) where a latent variable could be present in multiple factors. By combining Eq. (A.21) and Eq. (A.22) we can write an approximate posterior distribution related to the a'th node collection $p(Z|X) \approx \tilde{p}^{\{a\}}(Z|X)$ where

$$\begin{aligned} &\tilde{p}^{\{a\}}_{\Theta^{\{a\}}}(Z|X) = \\ &\quad p_{\Theta^{\{a\}},\mathrm{Pa}\check{\Theta}\left(C^{\{a\}}\right)} \left(Z^{\{a\}}|\mathrm{Pa}\check{Z}\left(C^{\{a\}}\right), X^{\{a\}} \right) \frac{p(X_G|Z^{\{A\}})}{\tilde{p}^{\{a\}}(X_G|X^{\{a\}})} \prod_{b\in A\setminus a} q_{\check{\Phi}^{\{b\}}} \left(Z^{\{b\}} \right) \end{aligned}$$

Where $\tilde{p}^{\{a\}}(X_G|X^{\{a\}})$ is defined in Eq. (A.26) in Appendix A.2. Based on Eq. (A.15) we can then define approximate sub-problems as

$$\min_{\Phi^{\{a\}}} D \left[q_{\Phi^{\{a\}}}(Z) || \tilde{p}^{\{a\}}_{\Theta^{\{a\}}}(Z|X) \right] \qquad \text{(A.23)}$$

Each of these sub-problems can then be solved successively or in parallel potentially on distributed compute instances as outlined in Algorithm A.1 and utilizing Stochastic variational inference as described in Section A.6.3. To see how this choice of factorization affects the posterior approximations and the learning of model parameters, Θ, consider the KL-divergence as divergence measure. Considering the KL-divergence we can rewrite the objective in Eq. (A.23) as shown in Eq. (A.27) through Eq. (A.28) in Appendix A.2 to obtain the following local dual objective for stochastic variational inference

$$\begin{aligned} L_{KL}^{\{a\}}\left(\Theta^{\{a\}}, \Phi^{\{a\}}\right) = & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(A.24)} \\ & E_{Z\sim\tilde{q}_{\text{Pa}\check{Z}}^{\{a\}}}\left[LogEvd_{X_G,\ X^{\{a\}}}^{\{a\}}\left(\Theta^{\{a\}}\right)\right] - D_{KL}\left[q_{\Phi^{\{a\}}}(Z)||\tilde{p}_{\Theta^{\{a\}}}^{\{a\}}(Z|X)\right] - C \end{aligned}$$

Where C is a constant with respect to $\Theta^{\{a\}}$ and $\Phi^{\{a\}}$, and $LogEvd^{\{a\}}\left(X_G,\ X^{\{a\}}\right)$ is the joint log-evidence over global observed variables, X_G, and observed variables, $X^{\{a\}}$, local to the a'th node collection. Since the first term on the right-hand side is constant with respect to $\Phi^{\{a\}}$, maximizing this local dual objective with respect to $\Phi^{\{a\}}$ will minimize the KL-divergence. Furthermore, since $D_{KL}\left[q_{\Phi^{\{a\}}}(Z)||\tilde{p}_{\Theta^{\{a\}}}^{\{a\}}(Z|X)\right] \geq 0$ by definition it follows from Eq. (A.24) that

$$E_{Z\sim\tilde{q}_{\text{Pa}\check{Z}}^{\{a\}}}\left[LogEvd^{\{a\}}\left(X_G,\ X^{\{a\}}\right)\right] - C \geq L_{KL}^{\{a\}}\left(\Theta^{\{a\}}, \Phi^{\{a\}}\right)$$

Therefore, by maximizing the local dual objective, $L_{KL}^{\{a\}}\left(\Theta^{\{a\}}, \Phi^{\{a\}}\right)$, with respect to the local model parameters, $\Theta^{\{a\}}$, we will push the expected joint log-evidence over the global, X_G, and the local, $X^{\{a\}}$, observed variables higher, where the expectation is taken with respect to the joint variational distribution over latent variables parent to the a'th node collection. In summary, this means that we can simultaneously fit our local model parameters, $\Theta^{\{a\}}$, to the evidence and obtain an approximate local posterior distribution, $q_{\Phi^{\{a\}}}(z^{\{a\}})$. While the above derivations were made for the KL-divergence, similar derivations can be done for the more general family of α-divergences.

To evaluate this local dual objective, we only need information related to the local node collection, its parents, and other node collections having the same global observed variables as children. Thereby providing substantially computational speedups for generative flow graphs with sparsely connected node collections and global observed variables. To use this procedure with a standard probabilistic programming language, we would have to create a probabilistic program fragment for each node collection, their corresponding variational distribution, and the global observed variables. These fragments would then have to be composed together to form the local objectives potentially in an automated fashion.

So far within this section, we have assumed that all sub-problems are solved through a variational problem as in Eq. (A.23). However, there are in principle no reasons why we could not use estimates of sub-posteriors, $q(z^{\{b\}})$, obtained through other means in Eq. (A.23), as long as we can sample from these sub-posteriors. Thus, making the outlined method very flexible to combine with other methods, albeit analysis of the results obtained through the combined inference becomes more difficult. It is also important to stress that the factorization used above is not unique. It would be interesting to investigate if other factorizations could be employed, and for which problems these factorizations could be useful.

In summary, if we can divide a generative flow graph representing an overall model of cognition into node collections and global observed variables, then we can utilize the combination of Message-Passing and Stochastic Variational inference presented within this section, to distribute the computational burden of performing inference within this model. At the same time, we can learn local model parameters. Thus, yielding a very flexible tool allowing us to fully specify the part of a model that we are certain about, and potentially learn the rest.

A.7 Probabilistic Programming Languages

So far, our focus has been on the representation of models defined by probabilistic programs, and on how to answer queries related to these models via modern probabilistic inference. However, we have not considered how this is made possible by probabilistic programming languages and their relation to deterministic programming languages. Here we will not give a detailed introduction to probabilistic programming and refer interested readers to other sources [19]' [38]' [39]. Instead, we will give a short overview of languages relevant to modeling cognition.

As already mentioned in Section A.5 the main characteristics of a probabilistic program is a construct for sampling randomly from distributions and another construct for condition values of variables in the program. The purpose of probabilistic programming languages is to provide these two constructs and to handle the underlying machinery for implementing inference algorithms and performing inference from these constructs. As with any other programming language, design decisions are not universally applicable or desirable, and different trade-offs are purposefully made to achieve different goals. This fact combined with theoretical advancements has resulted in several different probabilistic programming languages. For an extensive list see [19]. Some of these are domain-specific aimed at performing inference in a restricted class of probabilistic programs, such as STAN [40]. These restrictions are usually employed to obtain more efficient inference.

More interesting for the framework presented in Section A.3, however, are languages self-identifying as universal or general-purpose, such as Pyro [36] and Venture [41]' [42]. These languages aim at performing inference in arbitrary probabilistic programs. Thus maximizing the flexibility for modeling cognition.

A recent trend has been to build probabilistic programming languages on top of deep-learning libraries such as PyTorch [43] and TensorFlow [44]. This is done both to use the efficient tensor math, automatic differentiation, and hardware acceleration that these libraries provide and to get tighter integration of deep-learning models within probabilistic models. Examples of such languages are Pyro [36] and ProbTorch [45] build upon PyTorch, and Edward [46] build upon TensorFlow. Again, when considering the use within the framework presented in Section A.3, the languages based on PyTorch or TensorFlow 2.0 could potentially have the edge over others due to the dynamic approach to constructing computation graphs. This is because the dynamic computation graphs, more easily allow us to define dynamic models which include recursion and unbounded numbers of random choices [19]. Constructs potentially being indispensable for models of higher-level cognition supposed to evolve.

Python as the high-level general-purpose programming language it is makes modeling effortless in these languages. However, being based on python the computational efficiency of these languages is potentially limited by the need for interpretation. For this reason the relatively recent project called NumPyro [47] is under active development. NumPyro provides a backend to Pyro based on NumPy [48] and JAX [49] which enables just-in-time compilation, and thus potentially could provide much better computational efficiency, which again is essential for any practical robotic system.

To summarize, the choice of which probabilistic programming language to use depends on the flexibility needed to model cognition. However, universal or general-purpose languages based on deep-learning libraries possibly with just-in-time compilation and hardware acceleration seem promising for general modeling of cognition, and especially for cognitive robotics.

A.8 Application Examples

To demonstrate the concepts presented within this paper and the utility of the framework we have begun an initiative to implement some general applicable probabilistic programming idioms with basis in the "Standard Model of the Mind" [10] which is available as a GitHub repository [50]. The repository currently contain one such idiom called "__WM_planning_model(...)" implemented within the "Planning" class. The purpose of this idiom is to provide basic functionality to plan future actions of a robot based on cognitive con-

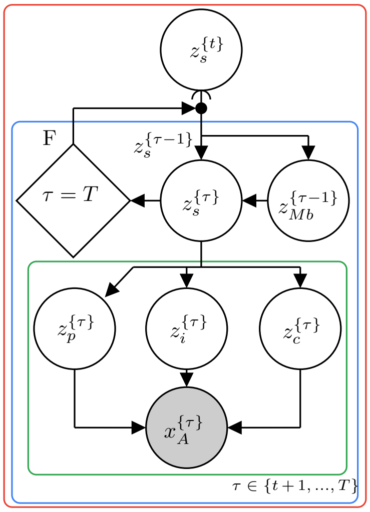


Fig. A.8: Generative Flow Graph for Planning
Excerpt of the generative flow graph representation of the idiom used in the abstract class "Planning" [50]. The red, green and blue colors relates node collections to the methods of the UML class diagram in Fig. A.9. The variables $z_p^{\{\tau\}}$, $z_i^{\{\tau\}}$, and $z_c^{\{\tau\}}$ represents progress, information gain, and constraints respectively. $z_s^{\{\tau\}}$ represents the robots internal state representation at time τ. $z_{Mb}^{\{\tau\}}$ represents the actions of the robot at time τ contained in the "motor buffer" (Mb). Finally, $x_A^{\{\tau\}}$ quantifies the amount of attention that the robot should give to a given state, $z_s^{\{\tau\}}$, through a weighting of the progress, information gain, and constraint variables.

cepts of desirability, progress, information gain, and constraints. Fig. A.8 illustrates an excerpt of the generative flow graph representation of the idiom. For an in dept presentation of the inner workings of the idiom we refer the reader to our other paper [51]. As seen from Fig. A.8, the idiom can be divided into a hierarchical structure of node collections, in which the red node collection internally depends on the blue node collection, that in turn depends on the green node collection and recursively on itself. Rather than implementing the idiom as one large probabilistic program this hierarchical structure allows us to implement the idiom as multiple smaller probabilistic programs. To keep the idiom generally applicable it is implemented within an abstract python class with a method for each of the node collections shown in Fig. A.8, that depends on some abstract methods that needs to be specified on a per application basis. Fig. A.9 shows a simplified UML class diagram of the main methods of the class. The method for each of the node collections in the idiom contains the main structure and functionality of the idiom. However, without the implementation of the abstract methods it is inoperative, and it is the implementation of e.g. the probabilistic program for the state transition, "p_z_s_tau(...)", that makes it application specific.

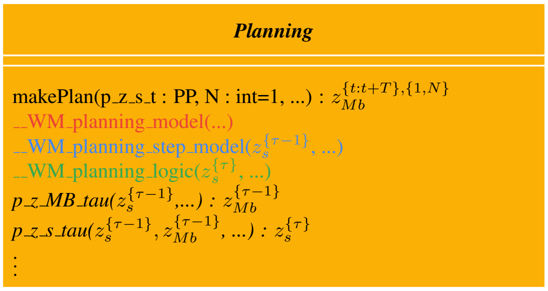

Planning
makePlan(p_z_s_t : PP, N : int=1, ...) : $z_{Mb}^{\{t:t+T\},\{1,N\}}$
__WM_planning_model(...)
__WM_planning_step_model($z_s^{\{\tau-1\}}$, ...)
__WM_planning_logic($z_s^{\{\tau\}}$, ...)
p_z_MB_tau($z_s^{\{\tau-1\}}$,...) : $z_{Mb}^{\{\tau-1\}}$
p_z_s_tau($z_s^{\{\tau-1\}}$, $z_{Mb}^{\{\tau-1\}}$, ...) : $z_s^{\{\tau\}}$
⋮

Fig. A.9: UML Class Diagram for "Planning"
Excerpt of the UML class diagram for the abstract class "Planning" [50]. For simplicity we have only included the methods relevant for the discussion within this manuscript. Italic text designates abstract classes and methods, "+" designates public methods, "PP" designates an argument of the type probabilistic program, and the red, green and blue color designates the methods implementing the node collections with corresponding colors in Fig. A.8.

In the simplest use-case the user can use the idiom simply by creating a child class that inherits the "Planning" class and implements the abstract methods. The user can then call the public method "makePlan(...)", which performs stochastic variational inference on the idiom and returns N samples from the approximate posterior

$$z_{Mb}^{\{t:t+T\},\{n\}} \sim q\left(z_{Mb}^{\{t:t+T\}}\right) \approx p\left(z_{Mb}^{\{t:t+T\}} | x_A^{\{t:t+T\}}\right) \tag{A.25}$$

constituting an optimal plan of future actions according to the idioms notion of progress, information gain, and constraints. The abstract methods that needs to be implemented are rather non-restrictive and most are only assumed to be probabilistic programs developed in Pyro [36] making the idiom very versatile. Besides the "Planning" class containing the idiom, the repository also contains applications examples for robot exploration and multi-robot navigation demonstrating different use-cases of idiom.

A.8.1 Robot Exploration

The purpose of this use-case is to demonstrate high-level robot motion planning with the goal of exploring an environment represented by a grid map in the long-term memory with a lidar mounted on a robot [51]. In this particular case the model of cognition aligned perfectly with the "__WM_planning_model(...)" idiom. Thus, the application where implemented simply as a child class implementing the abstract methods inherited from the abstract parent class "Planning" as illustrated in Fig. A.10. Thereby, the implementation for this application was greatly simplified.

In the related paper [51], the approach was tested on 35,126 2D floor plans available in the HouseExpo dataset utilising a modified version of the

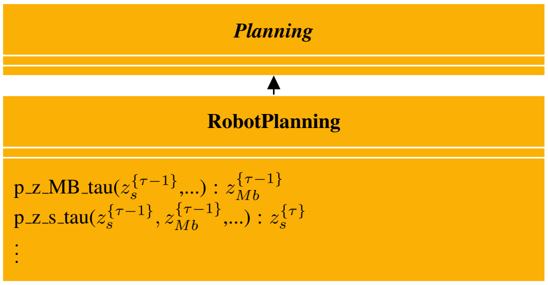

Planning
RobotPlanning
p_z_MB_tau($z_s^{\{\tau-1\}}$,...) : $z_{Mb}^{\{\tau-1\}}$
p_z_s_tau($z_s^{\{\tau-1\}}$, $z_{Mb}^{\{\tau-1\}}$,...) : $z_s^{\{\tau\}}$
⋮

Fig. A.10: UML Class Diagram for "robotPlanning"
Excerpt of the UML class diagram for the class "RobotPlanning" used for robot exploration [51]. The model for robot exploration aligned well with the idiom in Fig. A.8 and could thus be implemented as a child class inheriting from the abstract "Planning" class.

accompanying PseudoSLAM simulator [52]. Figure A.11 shows a snapshot of one of the simulations.

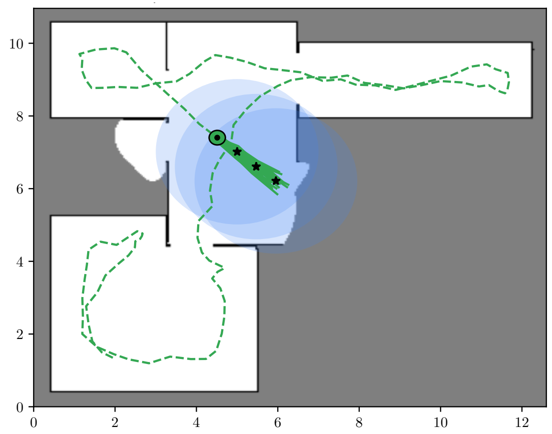


Fig. A.11: Robot Exploration Simulation
Results of a simulation of high-level robot motion planning with the goal of exploring an unknown environment with a lidar as the perceptual input. Gray indicate unexplored parts of the environment, white indicate unoccupied areas, black indicates obstacles, the green circle with a black boarder shows the current location of the robot, the green dashed line shows the robots past path, the solid green lines shows samples from the future optimal path distribution, the black stars shows the mean of these samples, and the transparent blue circles illustrates the lidars range at these positions.

From this extensive simulation study it was demonstrated that the method was indeed capable of planning actions to guide a robot towards new knowledge, thereby exploring a large part of most of the floor plans. During these simulations only $0,25$ ‰ of actions taken based on the "__WM_planning_model(...)" idiom resulted in collisions, thereby demonstrating the ability of the approach to avoide constraints. Currently, the implementation for this application uses down to approximately 1 s on planning depending on the settings. This is deemed sufficient for high-level planning in robotics applications and thus this simulation study also hints towards *Sufficient Efficiency* of the framework, which will only be corroborated by further code optimization.

A.8.2 Multi-robot Navigation

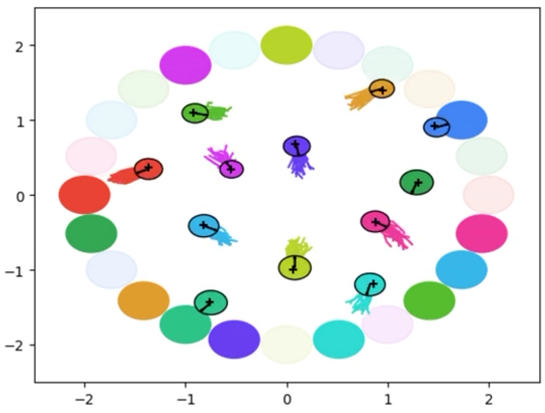


Fig. A.12: Multi-robot Navigation Simulation
Snapshot of a simulation with 12 robots utilizing the "Planning" idiom to plan actions towards their goal while avoiding collision with each other. Colored circles with a black boarder indicates the current location of the robots, solid colored lines indicates samples of their future planned path distribution, colored circles indicates their current goals, and transparent colored circles indicates their last goal.

The second application example relies heavily on the Stochastic Message-Passing approach described in Section A.6.4 to implement a simplistic form of communication between robots [26]. In this application N uni-cycle type robots has to plan low-level actions towards their goals while avoiding collisions with the other robots given knowledge about the other robots expected future path as illustrated in Fig. A.12. Figure A.13 shows a generative flow graph of the model derived for this problem [26].

By comparing Fig. A.13 with Fig. A.8, it is clear that the models are not exactly the same. However, the differences are encapsulated within the node collections marked by a green boarder in both diagrams. Thus, by creating a child class inheriting the "Planning" class, but overwriting the "__WM_planning_logic(...)" method it was possible to re-use a large part of the "__WM_planning_model(...)" idiom. Thereby, once more greatly simplifying the implementation process. Figure A.14 illustrates an excerpt of the UML class diagram used for this application. Notice here, that since the robots in

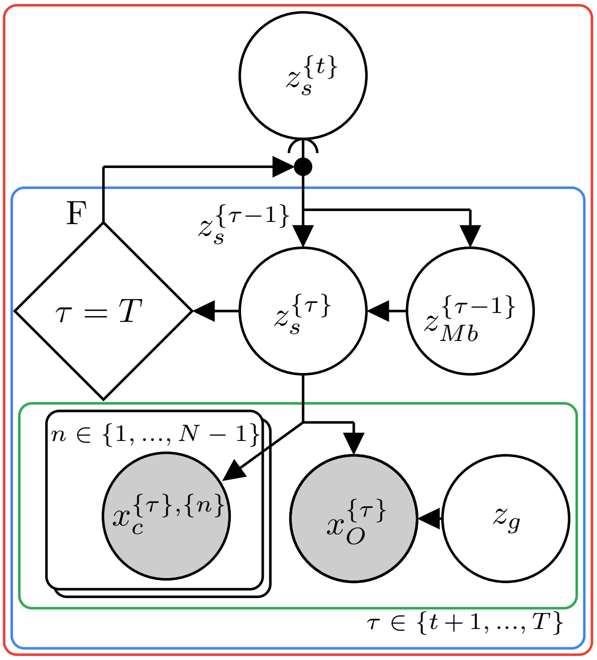


Fig. A.13: Generative Flow Graph for Multi-robot Navigation
Generative flow graph of the model derived for each of the robots in a multi-robot navigation problem [26]. $z_s^{\{\tau\}}$ represents the robots internal representation of its own state as well as the state of the other robots at time τ. $z_{Mb}^{\{\tau\}}$ represents the actions of the robot it self as well as the communicated planned actions of the other robots at time τ. $x_O^{\{\tau\}}$ quantifies how "optimal" the robots own state is in regards to getting closer to its own goal state, z_g. Finally, $x_c^{\{\tau\},\{n\}}$ represents the global constraints of avoiding collision with each of the $N-1$ other robots, i.e. X_G from Section A.6.4.

this application had to plan low level actions as well as keeping track of the state of the other robots the implementations of abstract methods like e.g. "p_z_MB_tau(...)" and "p_z_s_tau(...)" also had to be different from the ones used in the robot exploration application.

In the paper [26] related to this use-case, the approach used have been verified both through an extensive simulation study and a real-world experiment. From simulations of 2-32 robots it was concluded that the approach performs as well as, if not better than, the state-of-the-art algorithm B-UAVC [53] made exclusively for the problem of multi-robot collision avoidance. This was despite the fact that the above approach required way less analytical analysis, since only a relatively simple model of the problem had to be derived before the general concepts for performing inference in such a model presented within this paper could be applied. The approach was also tested in a real-world experiment with two TurtleBot3 Burger robots equipped with

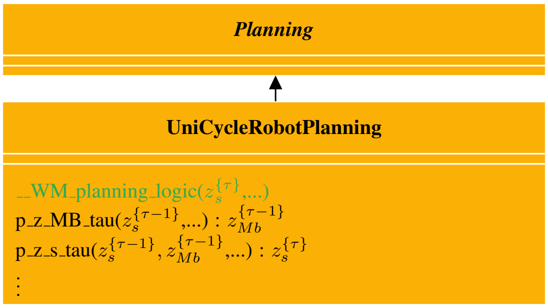

Planning
UniCycleRobotPlanning
__WM_planning_logic($z_s^{\{\tau\}}$,...)
p_z_MB_tau($z_s^{\{\tau-1\}}$,...) : $z_{Mb}^{\{\tau-1\}}$
p_z_s_tau($z_s^{\{\tau-1\}}, z_{Mb}^{\{\tau-1\}}$,...) : $z_s^{\{\tau\}}$
⋮

Fig. A.14: UML Class Diagram for "UniCycleRobotPlanning"
Excerpt of the UML class diagram for the class "UniCycleRobotPlanning" used for multi-robot navigation [26]. A large part of the model for multi-robot navigation aligned with the idiom in Fig. A.8 and could thus be implemented as a child class inheriting from the abstract "Planning" class overwriting the parts of the model that did not align.

an Intel NUC10FNK each for performing the necessary computations. The success of this real-world experiment demonstrated that sufficient computational efficiency is possible on standard hardware, as well as the real-world applicability of concepts presented within this paper.

A.8.3 Application Discussion

The point of the above examples is *NOT* that the method necessarily performs better than any other method or that the applications could not have been implemented in another way. The point is that by following the concepts of the framework presented in this manuscript it is possible to develop generally applicable models of cognition, that can easily be adapted and/or extended to new use-cases. Thereby, mitigating the complexity and burden of creating cognitive architectures for robotics application.

Although the repository currently do not contain a broad range of cognitive capabilities, the two examples demonstrates most of the concepts presented in Section A.5 through Section A.7. Specifically, the examples demonstrates the combined usage of probabilistic programming, inference in probabilistic programs, generative flow graphs, probabilistic programming idioms, and Stochastic message-passing for two real-world robotics applications.

Besides the present features of the idiom and "Planning" class, based on experiences from solving the multi-robot navigation problem the "__WM_planning_model(...)" idiom is currently being extended with a desirability variable for reaching goal states and detection of impasse. Currently, the "Planning" class makes use of stochastic variational inference. However, if we in the future want to use another algorithm for inference, we can simply inherit the "Planning" class and overwrite the "makePlan(...)" method to accommodate this inference algorithm. Since the idiom is implemented via the probabilistic programming language Pyro we do not need to re-implement the idiom it self to accommodate this inference algorithm. All of this together with the fact that the two vastly different applications are implemented from the same probabilistic programming idiom demonstrates how models developed in the proposed framework can encourage cooperation, re-use of existing results, and inspire new work.

A.9 Discussion

A.9.1 Conclusion

Inspired by Sigma's Cognitive Hourglass Model [1], we have outlined a framework for developing cognitive architectures for cognitive robotics. With probabilistic programs at the center, this framework is sufficiently general to span the full spectrum of emergent, symbolic, and hybrid architectures. By dividing cognitive architectures into a series of layers this framework provides levels of abstractions between models of cognition and the algorithms that implement them on computational devices. Some of these layers also directly relate to other fields of research, thereby encouraging better cooperation.

We also presented a graphical representation of probabilistic programs which we call generative flow graphs. We showed how such generative flow graphs can help identify important universal fragments of probabilistic programs and models. Fragments that could potentially be re-used in the development of other cognitive architectures. Thereby, again encouraging cooperation and easier re-use of existing results.

Furthermore, we introduced the problem of inference within probabilistic programs. We briefly reviewed possible approaches and argued that variational inference approaches seems interesting for cognitive robotics. We introduced two commonly used approaches called Message-Passing and Stochastic Variational Inference. We also outline the weaknesses of each approach and proposed a combined approach that we call Stochastic Message-Passing. The proposed approach provides a principle way of distributing the computational burden of inference and parameter learning.

To support implementations within the framework we reviewed existing probabilistic programming languages providing the necessary machinery to implement inference algorithms for and perform inference in probabilistic programs.

Finally, we provided a brief introduction to a initiative that both provide evidence for the applicability of the framework and concepts presented within this paper, but also functions as a starting point and a tool for researchers who wants to work within the framework.

The main topics within this paper have been the framework itself, the representation of cognitive models, and the computational burden. These topics are indeed essential ingredients of the framework and pose interesting research directions by themselves.

A.9.2 Limitations of the Study

The study presented within Section A.8 is limited to action selection through planning and control, and to fully demonstrate the flexibility of the proposed framework, applications to other cognitive tasks remain to be demonstrated. We, however, see no reasons why the same principles could not be applied to other aspects of cognition such as perception, attention, memory, social interaction, metacognition and even emotion.

A.10 Experimental Procedures

A.10.1 Resource Availability

Lead Contact

Requests for further information can be directed to the lead contact Malte Rørmose Damgaard at `mrd@es.aau.dk`

Materials Availability

This study did not generate new unique materials.

Data and Code Availability

This study did not generate new data or code. However, the data and code related to the two studies presented and discussed within Section A.8 are available online as a GitHub repository [50]. More specifically the data and code related to the robot exploration study presented in Section A.8.1 is available via Zenodo [54]. Similarely, the data and code related to the multi-robot exploration study presented in Section A.8.2 is available at the repository branch [55].

A.11 Author Contributions

Conceptualization, M.R.D., R.P., and T.B.; Investigation, M.R.D.; Methodology, M.R.D.; Formal Analysis, M.R.D.; Visualization, M.R.D.; Writing – Original Draft, M.R.D.; Writing – Review & Editing, M.R.D., R.P., and T.B.; Supervision, R.P., and T.B.

A.12 Declaration of interests

The authors declare no competing interests.

References

[1] P. S. Rosenbloom, A. Demski, and V. Ustun, "The sigma cognitive architecture and system: Towards functionally elegant grand unification," *Journal of Artificial General Intelligence*, vol. 7, no. 1, pp. 1–103, 2017, article in a periodical. [Online]. Available: https://doi.org/10.1515/jagi-2016-0001

[2] P. Haazebroek, S. van Dantzig, and B. Hommel, "A computational model of perception and action for cognitive robotics," *Cognitive Processing*, vol. 12, no. 4, p. 355–365, 2011, article in a periodical. [Online]. Available: https://doi.org/10.1007/s10339-011-0408-x

[3] J. Zhong, C. Ling, A. Cangelosi, A. Lotfi, and X. Liu, "On the gap between domestic robotic applications and computational intelligence," *Electronics*, vol. 10, no. 7, pp. 793:1–793:31, 2021, article in a periodical. [Online]. Available: https://www.mdpi.com/2079-9292/10/7/793

[4] I. Kotseruba and J. K. Tsotsos, "40 years of cognitive architectures: core cognitive abilities and practical applications," *Artificial Intelligence Review*, vol. 53, no. 1, pp. 17–94, 2020, article in a periodical. [Online]. Available: https://doi.org/10.1007/s10462-018-9646-y

[5] J. E. Laird, E. S. Yager, M. Hucka, and C. M. Tuck, "Robo-soar: An integration of external interaction, planning, and learning using soar," *Robotics and Autonomous Systems*, vol. 8, no. 1, pp. 113–129, 1991, article in a periodical. [Online]. Available: https://www.sciencedirect.com/science/article/pii/092188909190017F

[6] T. Huntsberger, "Envisioning cognitive robots for future space exploration," in *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 7710, Apr. 2010, article in proceedings.

[7] P. Bustos, J. Martínez-Gómez, I. García-Varea, L. Rodríguez-Ruiz, P. Bachiller, L. Calderita, L. Manso, A. Sánchez, A. Bandera, and J. Bandera, "Multimodal interaction with loki," in *Proceedings of Workshop of Physical Agents*, Sep. 2013, pp. 53–60, article in proceedings.

[8] A. Tanevska, F. Rea, G. Sandini, L. Cañamero, and A. Sciutti, "A socially adaptable framework for human-robot interaction," *Frontiers in Robotics and AI*, vol. 7, pp. 126:1–126:16, 2020, article in a periodical. [Online]. Available: https://www.frontiersin.org/article/10.3389/frobt.2020.00121

[9] P. Domingos and D. Lowd, *Markov Logic: An Interface Layer for Artificial Intelligence*, 1st ed. Morgan and Claypool Publishers, 2009, an entire book.

[10] J. E. Laird, C. Lebiere, and P. S. Rosenbloom, "A standard model of the mind: Toward a common computational framework across artificial intelligence, cognitive science, neuroscience, and robotics," *AI Magazine*, vol. 38, no. 4, pp. 13–26, Dec. 2017, article in a periodical. [Online]. Available: https://ojs.aaai.org/index.php/aimagazine/article/view/2744

[11] Steve Deering, "Watching the waist of the protocol hourglass," Keynote at ICNP '98, https://ant.isi.edu/csci551/images/3/32/Deering98a.pdf, Oct. 1998, keynote.

[12] M. Fadlil, K. Ikeda, K. Abe, T. Nakamura, and T. Nagai, "Integrated concept of objects and human motions based on multi-layered multimodal lda," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 2256–2263, article in proceedings.

[13] A. Taniguchi, Y. Hagiwara, T. Taniguchi, and T. Inamura, "Improved and scalable online learning of spatial concepts and language models with mapping," *Autonomous Robots*, vol. 44, pp. 927–946, Jul. 2020, article in a periodical. [Online]. Available: https://doi.org/10.1007/s10514-020-09905-0

[14] ——, "Spatial concept-based navigation with human speech instructions via probabilistic inference on bayesian generative model," *Advanced Robotics*, vol. 34, no. 19, pp. 1213–1228, 2020, article in a periodical. [Online]. Available: https://doi.org/10.1080/01691864.2020.1817777

[15] K. Miyazawa, T. Horii, T. Aoki, and T. Nagai, "Integrated cognitive architecture for robot learning of action and language," *Frontiers in Robotics and AI*, vol. 6, pp. 1–20, 2019, article in a periodical. [Online]. Available: https://www.frontiersin.org/article/10.3389/frobt.2019.00131

[16] T. Nakamura, T. Nagai, and T. Taniguchi, "Serket: An architecture for connecting stochastic models to realize a large-scale cognitive model," *Frontiers in Neurorobotics*, vol. 12, pp. 25:1–25:16, 2018, article in a

periodical. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnbot.2018.00025

[17] T. Taniguchi, T. Nakamura, M. Suzuki, R. Kuniyasu, K. Hayashi, A. Taniguchi, T. Horii, and T. Nagai, "Neuro-serket: Development of integrative cognitive system through the composition of deep probabilistic generative models," *New Gen. Comput.*, vol. 38, no. 1, p. 23–48, Mar. 2020, article in a periodical. [Online]. Available: https://doi.org/10.1007/s00354-019-00084-w

[18] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001, article in a periodical.

[19] J.-W. van de Meent, B. Paige, H. Yang, and F. Wood, "An introduction to probabilistic programming," Preprint at arXiv, 2018, article on a preprint server or other repository.

[20] H. A. Simon, "Rational choice and the structure of the environment," *Psychological review*, vol. 63, no. 2, pp. 129–138, 1956, article in a periodical.

[21] A. D. Gordon, T. A. Henzinger, A. V. Nori, and S. K. Rajamani, "Probabilistic programming," ser. FOSE 2014, vol. 1. Association for Computing Machinery, 2014, pp. 167–181, article in proceedings. [Online]. Available: https://doi.org/10.1145/2593882.2593900

[22] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009, an entire book.

[23] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE Robotics Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006, article in a periodical.

[24] S. Levine, "Reinforcement learning and control as probabilistic inference: Tutorial and review," Preprint at arXiv, 2018, article on a preprint server or other repository. [Online]. Available: http://arxiv.org/abs/1805.00909

[25] C. Zhang, J. Bütepage, H. Kjellström, and S. Mandt, "Advances in variational inference," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 8, pp. 2008–2026, 2019, article in a periodical.

[26] M. R. Damgaard, R. Pedersen, and T. Bak, "Study of variational inference for flexible distributed probabilistic robotics," *Robotics*, vol. 11, no. 2, pp. 38:1–38:19, 2022, article in a periodical. [Online]. Available: https://www.mdpi.com/2218-6581/11/2/38

[27] T. Minka, "Divergence measures and message passing," https://www.microsoft.com/en-us/research/publication/divergence-measures-and-message-passing/, Microsoft, Tech. Rep. MSR-TR-2005-173, Jan. 2005. [Online]. Available: https://www.microsoft.com/en-us/research/publication/divergence-measures-and-message-passing/

[28] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic variational inference," *Journal of Machine Learning Research*, vol. 14, no. 4, pp. 1303–1347, 2013, article in a periodical. [Online]. Available: http://jmlr.org/papers/v14/hoffman13a.html

[29] R. Ranganath, S. Gerrish, and D. Blei, "Black Box Variational Inference," in *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, S. Kaski and J. Corander, Eds., vol. 33. Reykjavik, Iceland: PMLR, Apr. 2014, pp. 814–822, article in proceedings. [Online]. Available: http://proceedings.mlr.press/v33/ranganath14.html

[30] A. Kucukelbir, D. Tran, R. Ranganath, A. Gelman, and D. M. Blei, "Automatic differentiation variational inference," *Journal of Machine Learning Research*, vol. 18, no. 14, pp. 1–45, 2017, article in a periodical. [Online]. Available: http://jmlr.org/papers/v18/16-107.html

[31] Y. Li and R. E. Turner, "Rényi divergence variational inference," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016, pp. 1–9, article in proceedings. [Online]. Available: https://proceedings.neurips.cc/paper/2016/file/7750ca3559e5b8e1f44210283368fc16-Paper.pdf

[32] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., Apr. 2014, pp. 1–14, article in proceedings. [Online]. Available: http://arxiv.org/abs/1312.6114

[33] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3, pp. 229–256, 1992, article in a periodical. [Online]. Available: https://doi.org/10.1007/BF00992696

[34] T. Salimans and D. A. Knowles, "Fixed-form variational posterior approximation through stochastic linear regression," *Bayesian Analysis*, vol. 8, no. 4, p. 837–882, Dec. 2013, article in a periodical. [Online]. Available: http://dx.doi.org/10.1214/13-BA858

[35] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ser. ICML'14. JMLR.org, 2014, pp. II–1278–II–1286, article in proceedings.

[36] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. A. Szerlip, P. Horsfall, and N. D. Goodman, "Pyro: Deep universal probabilistic programming," *J. Mach. Learn. Res.*, vol. 20, pp. 28:1–28:6, 2019, article in a periodical. [Online]. Available: http://jmlr.org/papers/v20/18-403.html

[37] J. Zhang, P. Raman, S. Ji, H.-F. Yu, S. Vishwanathan, and I. Dhillon, "Extreme stochastic variational inference: Distributed inference for large scale mixture models," in *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and M. Sugiyama, Eds., vol. 89. PMLR, Apr. 2019, pp. 935–943, article in proceedings. [Online]. Available: https://proceedings.mlr.press/v89/zhang19c.html

[38] C. Davidson-Pilon, *Bayesian Methods for Hackers: Probabilistic Programming and Bayesian Inference*, 1st ed. Addison-Wesley Professional, 2015, an entire book.

[39] A. Pfeffer, *Practical Probabilistic Programming*, 1st ed. USA: Manning Publications Co., 2016, an entire book.

[40] B. Carpenter, A. Gelman, M. D. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. Brubaker, J. Guo, P. Li, and A. Riddell, "Stan: A probabilistic programming language," *Journal of Statistical Software*, vol. 76, no. 1, p. 1–32, 2017, article in a periodical. [Online]. Available: https://www.jstatsoft.org/index.php/jss/article/view/v076i01

[41] V. Mansinghka, D. Selsam, and Y. Perov, "Venture: a higher-order probabilistic programming platform with programmable inference," Preprint at arXiv, 2014, article on a preprint server or other repository.

[42] V. K. Mansinghka, U. Schaechtle, S. Handa, A. Radul, Y. Chen, and M. Rinard, "Probabilistic programming with programmable inference," *SIGPLAN Not.*, vol. 53, no. 4, p. 603–616, Jun. 2018, article in a periodical. [Online]. Available: https://doi.org/10.1145/3296979.3192409

[43] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative

style, high-performance deep learning library." in *NeurIPS*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. B. Fox, and R. Garnett, Eds., 2019, pp. 8024–8035, article in proceedings. [Online]. Available: http://dblp.uni-trier.de/db/conf/nips/nips2019.html#PaszkeGMLBCKLGA19

[44] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: A system for Large-Scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. Savannah, GA: USENIX Association, Nov. 2016, pp. 265–283, article in proceedings. [Online]. Available: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi

[45] N. Siddharth, B. Paige, J.-W. van de Meent, A. Desmaison, N. D. Goodman, P. Kohli, F. Wood, and P. H. Torr, "Learning disentangled representations with semi-supervised deep generative models," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, pp. 5927–5937, article in proceedings.

[46] D. Tran, M. D. Hoffman, R. A. Saurous, E. Brevdo, K. Murphy, and D. M. Blei, "Deep probabilistic programming," Preprint at arXiv, 2017, article on a preprint server or other repository. [Online]. Available: https://arxiv.org/abs/1701.03757

[47] D. Phan, N. Pradhan, and M. Jankowiak, "Composable effects for flexible and accelerated probabilistic programming in numpyro," *CoRR*, vol. abs/1912.11554, pp. 1–10, 2019, article in a periodical. [Online]. Available: http://arxiv.org/abs/1912.11554

[48] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020, article in a periodical. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[49] R. Frostig, M. J. Johnson, and C. Leary, "Compiling machine learning programs via high-level tracing," *Systems for Machine Learning*, pp. 23–24, 2018, article in a periodical.

[50] M. R. Damgaard, "ProbMind," GitHub repository, https://github.com/damgaardmr/probMind, 2022, software available online.

[51] M. R. Damgaard, R. Pedersen, and T. Bak, "A probabilistic programming idiom for active knowledge search," Preprint at arXiv, 2022, article on a preprint server or other repository.

[52] T. Li, D. Ho, C. Li, D. Zhu, C. Wang, and M. Q.-H. Meng, "Houseexpo: A large-scale 2d indoor layout dataset for learning-based algorithms on mobile robots," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5839–5846, article in proceedings.

[53] H. Zhu, B. Brito, and J. Alonso-Mora, "Decentralized probabilistic multi-robot collision avoidance using buffered uncertainty-aware voronoi cells," *Autonomous Robots*, vol. 46, pp. 401–420, 2022, article in a periodical.

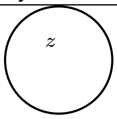
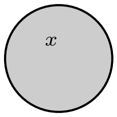
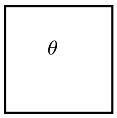
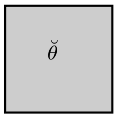
[54] M. R. Damgaard, "probmind release for "a probabilistic programming idiom for active knowledge search"," Zenodo Archive, https://doi.org/10.5281/zenodo.5841292, Jan. 2022, software available online. [Online]. Available: https://doi.org/10.5281/zenodo.5841292

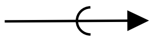
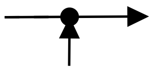
[55] ——, "Multi robot planning simulation," GitHub repository, https://github.com/damgaardmr/probMind/tree/d0ba27687b373ff04eb790ef38b21ca8572d8c8a/examples/multiRobotPlanning, Jan. 2022, software available online.

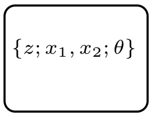
Appendix A Supplemental Experimental Procedures

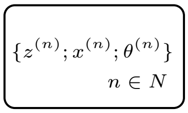
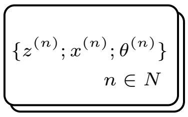
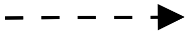
Appendix A.1 Generative Flow Graphs

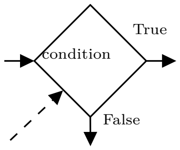
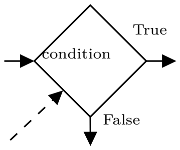
Table A.1: Semantics of the Generative Flow Graph representation of probabilistic programs.

Symbol	Description	Meaning
	Circle without colored background	A node symbolizing a probabilistic variable, corresponding to a "*sample*" function or keyword in the probabilistic program. We denote this as a "*Latent Variable Node*".
	Circle with colored background	A node symbolizing an observed probabilistic variable, corresponding to a "*observe*" function or keyword in the probabilistic program. We denote this as an "*Observed Variable Node*".
	Square without colored background	A node symbolizing learn-able parameters in the probabilistic program. That is model parameters that can change at run-time. We denote this as a "*Variable Parameter Node*"
	Square with colored background	A node symbolizing fixed parameters in the probabilistic program. Use-full for representing parameters that cannot change at run-time such as tuning parameters. We denote this as a "*Fixed parameter node*"
$\longrightarrow$	simple arrow	Link showing the generative path in a Probabilistic Program. The arrow can start in Latent Variable Nodes and Parameter nodes, and only point towards *latent variable nodes* and *observed variable nodes*. In large graphs or in cases where the origin of a link can be uncertain the readability can be improved by adding the name of the node from which the link originates next to the link. We denote this as a "*Generative Link*"

	Half circle on arrow	Symbolizing that operations downstream of this link in the probabilistic program should not influence nodes upstream to this link in the generative path. I.e. information such as accumulated gradients in a backward pass of automatic differentiation should not be propagated back through this link, corresponding to a ".detach()" and ".stop_gradient" call in PyTorch and TensorFlow, respectively. We denote this as a "*Detached Link*". A "variable parameter node" having only a detached link, can be considered a "fixed parameter node" or vice versa.
	Simple arrow pointing towards another simple arrow	Depending on the context, samples from different *latent variable nodes* may be used to generate the next sample. The generative link from the most recent sampled *latent variable node* in the generative path is the new active link. Used, e.g., to represent for loops.

	Polygon with rounded corners	Collection of nodes with internal dependency structure. The full structure of links between nodes can be shown inside the polygon. Alternatively, the names of the nodes can be written between curly brackets, $\{z; x; \theta\}$, with a semi-colon separating variables nodes, z, observed nodes, x, and parameter nodes, θ, in that order. Finally, a node collection can also simply be defined somewhere else, $C = \{z; x; \theta\}$, and referred to by, e.g., a single letter, in which case we encourage the use of capital letters to emphasize the difference from the other types of nodes. In cases where one or more types of nodes are not present in a collection, both semi-colons should still be there. E.g., $\{; x_1, x_2; \}$ for a collection the two observed nodes x_1 and x_2. Such a collection of nodes directly corresponds to the factor $p_\theta(x_1, x_2\|z)$ in a factorization of the joint distribution over all variables in a model. We denote this as a "*Node Collection*". A node is only allowed to be within one node collection unless it is within a node collection fully nested within another.

	Index specification in one corner of a polygon with rounded corners	Collection of nodes with indexed names. When the index is used to name nodes it is good practice to use the index in a superscript and encapsulate it in round brackets to emphasize that it is an index. We denote such a node collection an "*Indexed Node Collection*". The valid index should be clear from the context. E.g., in the case of a loop around the indexed collection, the index is incremented each time the loop enters the indexed node collection. When no such loops exist around the indexed node collection that could potentially cause ambiguity, it can be used instead of writing all the variables in a node collection with the curly brackets. For an example see Fig. A.4c
	Stacked polygons with rounded corners	Explicitly representation of multiple identical collections of nodes conditionally independent given their parents or simply independent if there are no parents. Here the one index is used for each of the independent collections.
	Dashed arrow	A link symbolizing an indirect relation between nodes. Such a link is non-generative, meaning that it is not directly used as a parameter in the generation of other samples, but only influences the generative path of other samples. We denote this as an "*Influence Link*"

	A Polygon with generative links connected at vertices, influence links towards the polygon connected at edges, and conditional values nearby generative links pointing away from the polygon	Node representing a condition changing the "direction" of the generative flow in a probabilistic program. We denote this as "*conditioned generative branching*".
	A Polygon with generative links connected at vertices, influence links towards the polygon connected at edges, and conditional values nearby generative links pointing towards the polygon	Node representing a condition selecting one out of two or more possible generative flows from parent nodes. We denote this as "*conditioned generative selection*".

Appendix A.2 Rewriting KL-divergence for Stochastic Message-Passing

In this section, we will derive the dual objective for the combination of message-passing and stochastic variational inference presented in Section A.6.4. Start by considering

$$
\begin{aligned}
p_\Theta(X_G|X^{\{A\}}) &= \int p_\Theta(X_G, Z^{\{A\}}|X^{\{A\}})dZ \\
&= \int p(X_G|Z^{\{A\}}, X^{\{A\}})p_\Theta(Z^{\{A\}}|X^{\{A\}})dZ \\
&= \int p(X_G|Z^{\{A\}})p_\Theta(Z^{\{A\}}|X^{\{A\}})dZ \\
&= \int p(X_G|Z^{\{A\}}) \prod_{a\in A} p_{\Theta^{\{a\}},\mathrm{Pa}\check{\Theta}\left(C^{\{a\}}\right)}\left(Z^{\{a\}}|\mathrm{Pa}\check{Z}\left(C^{\{a\}}\right), X^{\{a\}}\right) dZ
\end{aligned}
$$

By replacing $p_{\Theta^{\{b\}},\mathrm{Pa}\check{\Theta}\left(C^{\{b\}}\right)}\left(Z^{\{b\}}|\mathrm{Pa}\check{Z}\left(C^{\{b\}}\right), X^{\{b\}}\right)$ with their corresponding variational distributions $q_{\check{\Phi}^{\{b\}}}\left(Z^{\{b\}}\right)$ for $b \in A \setminus a$ we obtain

$$
\begin{aligned}
&\tilde{p}^{\{a\}}(X_G|X^{\{a\}}) = \\
&\quad \int p(X_G|Z^{\{A\}})p_{\Theta^{\{a\}},\mathrm{Pa}\check{\Theta}\left(C^{\{a\}}\right)}\left(Z^{\{a\}}|\mathrm{Pa}\check{Z}\left(C^{\{a\}}\right), X^{\{a\}}\right) \prod_{b\in A\setminus a} q_{\check{\Phi}^{\{b\}}}\left(Z^{\{b\}}\right) dZ
\end{aligned}
\tag{A.26}
$$

where we have used $\tilde{p}^{\{a\}}(X_G|X^{\{a\}})$ instead of $\tilde{p}^{\{a\}}(X_G|X^{\{A\}})$ to emphasize the conditional independence between X_G and $X^{\{b\}}$ given $Z^{\{b\}}$ for $b \in A \setminus a$ implicitly assumed by the approximation. Furthermore, also notice that we can rewrite the distribution, $p(X_G|Z^{\{A\}})$, as follows

$$
\begin{aligned}
p(X_G|Z^{\{A\}}) &= p(\mathrm{Ch}X_G\left(Z^{\{a\}}\right)|Z^{\{A\}})p(X_G \setminus \mathrm{Ch}X_G\left(Z^{\{a\}}\right)|Z^{\{A\}}) \\
&= p\left(\mathrm{Ch}X_G\left(Z^{\{a\}}\right)|\mathrm{Pa}Z\left(\mathrm{Ch}X_G\left(Z^{\{a\}}\right)\right)\right) \\
&\qquad \cdot p\left(X_G \setminus \mathrm{Ch}X_G\left(Z^{\{a\}}\right)|Z^{\{A\}} \setminus Z^{\{a\}}\right)
\end{aligned}
$$

by separating the global observed variables, X_G, into those who are direct children of $Z^{\{a\}}$, and those who are not. With the definition above and the ones given in Section A.6.4, we can rewrite the KL-divergence as follows:

$$
D_{KL}\left[q_{\Phi^{\{a\}}}(Z)||\tilde{p}^{\{a\}}_{\Theta^{\{a\}}}(Z|X)\right] \tag{A.27}
$$

$$= \int_Z q_{\Phi^{\{a\}}}(Z) \log \left(\frac{q_{\Phi^{\{a\}}}(Z)}{\tilde{p}^{\{a\}}_{\Theta^{\{a\}}}(Z|X)} \right) dZ$$

$$= \int_Z q_{\Phi^{\{a\}}}(Z) \log \left(\frac{q_{\Phi^{\{a\}}}(Z^{\{a\}}) \prod_{b \in A \setminus a} q_{\breve{\Phi}^{\{b\}}}(Z^{\{b\}})}{\left[\begin{array}{l} \frac{p(X_G|Z^{\{A\}})}{\tilde{p}^{\{a\}}(X_G|X^{\{a\}})} \prod_{b \in A \setminus a} q_{\breve{\Phi}^{\{b\}}} \left(Z^{\{b\}} \right) \\ \cdot p_{\Theta^{\{a\}}, \mathrm{Pa}\breve{\Theta}(C^{\{a\}})} \left(Z^{\{a\}} | \mathrm{Pa}\check{Z} \left(C^{\{a\}} \right), X^{\{a\}} \right) \end{array} \right]} \right) dZ$$

$$= \int_Z q_{\Phi^{\{a\}}}(Z) \log \left(\frac{q_{\Phi^{\{a\}}}(Z^{\{a\}})}{\left[\begin{array}{l} \frac{p(X_G|Z^{\{A\}})}{\tilde{p}^{\{a\}}(X_G|X^{\{a\}})} \\ \cdot p_{\Theta^{\{a\}}, \mathrm{Pa}\breve{\Theta}(C^{\{a\}})} \left(Z^{\{a\}} | \mathrm{Pa}\check{Z} \left(C^{\{a\}} \right), X^{\{a\}} \right) \end{array} \right]} \right) dZ$$

$$= \int_Z q_{\Phi^{\{a\}}}(Z) \log \left(\frac{q_{\Phi^{\{a\}}} \left(Z^{\{a\}} \right)}{\frac{p(X_G|Z^{\{A\}})}{\tilde{p}^{\{a\}}(X_G|X^{\{a\}})} \frac{p_{\Theta^{\{a\}}, \mathrm{Pa}\breve{\Theta}(C^{\{a\}})}(Z^{\{a\}}, X^{\{a\}} | \mathrm{Pa}\check{Z}(C^{\{a\}}))}{p_{\Theta^{\{a\}}, \mathrm{Pa}\breve{\Theta}(C^{\{a\}})}(X^{\{a\}} | \mathrm{Pa}\check{Z}(C^{\{a\}}))}} \right) dZ$$

$$= \int_Z q_{\Phi^{\{a\}}}(Z) \log \left(\frac{q_{\Phi^{\{a\}}}(Z^{\{a\}})}{\begin{array}{c} \left[\frac{\begin{array}{l} p(\mathrm{Ch}X_G(Z^{\{a\}}) | \mathrm{Pa}Z(\mathrm{Ch}X_G(Z^{\{a\}}))) \\ \cdot p \left(X_G \setminus \mathrm{Ch}X_G \left(Z^{\{a\}} \right) | Z^{\{A\}} \setminus Z^{\{a\}} \right) \end{array}}{\tilde{p}^{\{a\}}(X_G|X^{\{a\}})} \right] \\ \cdot \frac{p_{\Theta^{\{a\}}, \mathrm{Pa}\breve{\Theta}(C^{\{a\}})}(Z^{\{a\}}, X^{\{a\}} | \mathrm{Pa}\check{Z}(C^{\{a\}}))}{p_{\Theta^{\{a\}}, \mathrm{Pa}\breve{\Theta}(C^{\{a\}})}(X^{\{a\}} | \mathrm{Pa}\check{Z}(C^{\{a\}}))} \end{array}} \right) dZ$$

$$\begin{aligned} = \int_Z q_{\Phi^{\{a\}}}(Z) \log \left(\frac{q_{\Phi^{\{a\}}}(Z^{\{a\}})}{\left[\begin{array}{l} p \left(\mathrm{Ch}X_G \left(Z^{\{a\}} \right) | \mathrm{Pa}Z \left(\mathrm{Ch}X_G \left(Z^{\{a\}} \right) \right) \right) \\ \cdot p_{\Theta^{\{a\}}, \mathrm{Pa}\breve{\Theta}(C^{\{a\}})} \left(Z^{\{a\}}, X^{\{a\}} | \mathrm{Pa}\check{Z} \left(C^{\{a\}} \right) \right) \end{array} \right]} \right) dZ \\ + \int_Z q_{\Phi^{\{a\}}}(Z) \log \left(\begin{array}{l} \tilde{p}^{\{a\}}(X_G|X^{\{a\}}) \\ \cdot p_{\Theta^{\{a\}}, \mathrm{Pa}\breve{\Theta}(C^{\{a\}})} \left(X^{\{a\}} | \mathrm{Pa}\check{Z} \left(C^{\{a\}} \right) \right) \end{array} \right) dZ \\ - \int_Z q_{\Phi^{\{a\}}}(Z) \log \left(p \left(X_G \setminus \mathrm{Ch}X_G \left(Z^{\{a\}} \right) | Z^{\{A\}} \setminus Z^{\{a\}} \right) \right) dZ \end{aligned}$$

$$= \underbrace{E_{Z \sim \tilde{q}^{\{a\}}_{\Phi^{\{a\}}}} \left[\log \left(\frac{q_{\Phi^{\{a\}}} \left(Z^{\{a\}} \right)}{\begin{array}{l} p \left(\mathrm{Ch}X_G \left(Z^{\{a\}} \right) | \mathrm{Pa}Z \left(\mathrm{Ch}X_G \left(Z^{\{a\}} \right) \right) \right) \\ \cdot p_{\Theta^{\{a\}}, \mathrm{Pa}\breve{\Theta}(C^{\{a\}})} \left(Z^{\{a\}}, X^{\{a\}} | \mathrm{Pa}\check{Z} \left(C^{\{a\}} \right) \right) \end{array}} \right) \right]}_{-L^{\{a\}}_{KL}(\Theta^{\{a\}}, \Phi^{\{a\}})}$$

$$+ E_{Z\sim\tilde{q}^{\{a\}}_{\mathrm{Pa}\check{Z}}}\left[\underbrace{\log\left(\tilde{p}^{\{a\}}(X_G|X^{\{a\}})p_{\Theta^{\{a\}},\mathrm{Pa}\check{\Theta}\left(C^{\{a\}}\right)}\left(X^{\{a\}}|\mathrm{Pa}\check{Z}\left(C^{\{a\}}\right)\right)\right)}_{LogEvd^{\{a\}}_{X_G,\ X^{\{a\}}}\left(\Theta^{\{a\}}\right)}\right]$$

(A.28)

$$\underbrace{- E_{Z\sim\prod_{b\in A\setminus a}q_{\check{\Phi}^{\{b\}}}\left(Z^{\{b\}}\right)}\left[\log\left(p\left(X_G\setminus \mathrm{Ch}X_G\left(Z^{\{a\}}\right)|Z^{\{A\}}\setminus Z^{\{a\}}\right)\right)\right]}_{C}$$

where

$$\tilde{q}^{\{a\}}_{\Phi^{\{a\}}} = q_{\Phi^{\{a\}}}\left(Z^{\{a\}}\right)\prod_{Z^{\{b\}}\in\mathrm{Pa}\check{Z}\left(C^{\{a\}}\right)\cup\mathrm{Pa}\check{Z}\left(\mathrm{Ch}X_G\left(Z^{\{a\}}\right)\right)\setminus Z^{\{a\}}} q_{\check{\Phi}^{\{b\}}}\left(Z^{\{b\}}\right)$$

is the joint variational distribution over the latent variables, $Z^{\{a\}}$, local to the a'th node collection, and the latent variables parent to the a'th node collection, $\mathrm{Pa}\check{Z}\left(C^{\{a\}}\right)$, or having the same child global observed variables as the a'th node collection, $Z^{\{b\}}\in\mathrm{Pa}\check{Z}\left(\mathrm{Ch}X_G\left(Z^{\{a\}}\right)\right)\setminus Z^{\{a\}}$. Furthermore,

$$\tilde{q}^{\{a\}}_{\mathrm{Pa}\check{Z}} = \prod_{Z^{\{b\}}\in\mathrm{Pa}\check{Z}\left(C^{\{a\}}\right)} q_{\check{\Phi}^{\{b\}}}\left(Z^{\{b\}}\right)$$

is the joint variational distribution over the latent variables parent to the a'th node collection, $\mathrm{Pa}\check{Z}\left(C^{\{a\}}\right)$. Finally, $LogEvd^{\{a\}}_{X_G,\ X^{\{a\}}}\left(\Theta^{\{a\}}\right)$ denotes the joint log-evidence for X_G and $X^{\{a\}}$, and C is constant with respect to $\Theta^{\{a\}},\Phi^{\{a\}}$. By simple rearranging terms we obtain the dual objective

$$\begin{aligned}L^{\{a\}}_{KL}\left(\Theta^{\{a\}},\Phi^{\{a\}}\right) =& E_{Z\sim\tilde{q}^{\{a\}}_{\mathrm{Pa}\check{Z}}}\left[LogEvd^{\{a\}}_{X_G,\ X^{\{a\}}}\left(\Theta^{\{a\}}\right)\right]\\ &- D_{KL}\left[q_{\Phi^{\{a\}}}(Z)||\tilde{p}^{\{a\}}_{\Theta^{\{a\}}}(Z|X)\right] - C\end{aligned}$$

Paper B

Study of Variational Inference for Flexible Distributed Probabilistic Robotics

Malte Rørmose Damgaard, Rasmus Pedersen, and Thomas Bak

The paper has been published in
Robotics Volume 11, Issue 2, 38, 2022.
`https://doi.org/10.3390/robotics11020038`

Abstract

By combining stochastic variational inference with message passing algorithms, we show how to solve the highly complex problem of navigation and avoidance in distributed multi-robot systems in a computationally tractable manner, allowing online implementation. Subsequently, the proposed variational method lends itself to more flexible solutions than prior methodologies. Furthermore, the derived method is verified both through simulations with multiple mobile robots and a real world experiment with two mobile robots. In both cases, the robots share the operating space and need to cross each other's paths multiple times without colliding.

B.1 Introduction

Uncertainty is an inherent part of robotics that must be dealt with explicitly through the robust design of sensors, mechanics, and algorithms. Unlike many other engineering research areas that also have to deal with uncertainties, robotics problems usually also consist of a heterogeneous set of interconnected sub-problems and have strict real-time requirements, making it even harder to deal with uncertainty in an appropriate manner [1].

A common approach to model uncertainties in robotics is to employ probability mass functions and/or probability density functions, hereinafter jointly referred to as probability distributions, over model variables. One can then represent many classical robotics problems as a joint distribution, $p(x, z)$, over observable variables, x, and latent variables, z. Given the knowledge that the observable variables, x, can be assigned specific values $\overline{x}$, solving the problem then boils down to solving the posterior inference problem given by the conditional distribution

$$p(z|x = \overline{x}) = \frac{p(x = \overline{x}, z)}{p(x = \overline{x})} \tag{B.1}$$

$$= \frac{p(x = \overline{x}, z)}{\int p(x = \overline{x}, z)\, dz}. \tag{B.2}$$

Unfortunately, the marginalization by the integral in the denominator of Equation (B.2) is, in general, intractable to compute in most realistic problems, and thereby the reason why one often has to resort to approximate inference [2].

The classical solution to this problem has been to simplify the model of a problem, p, sufficiently to obtain an approximate problem definition, $q \approx p$, for which one can derive or use analytical solutions such as the Kalman filter [3], henceforth referred to as the "model simplification method". Typically, it is only possible to derive analytical solutions for a very limited set of probability distributions. Thereby, it may be necessary to apply crude approximations to obtain a solution, making it a rather inflexible method. However, such solutions tend to be computationally efficient, which is why they were commonly used in the early days of probabilistic robotics where computational resources were limited. One good example of this is Kalman filter-based simultaneous localization and mapping (SLAM). It is well known that in many cases the true posterior, p, is multi-modal, e.g., due to ambiguities and changes in the environment [4]. However, Kalman filter-based SLAM implicitly assumes a uni-modal Gaussian posterior, q, which in some cases can lead to poor solutions.

Another possibility is to use Monte Carlo methods such as particle filters. These methods have the benefit that they usually do not enforce any restrictions on the model, p, making these methods highly flexible. Furthermore, with these methods, it is often possible to obtain any degree of accuracy at the cost of losing computational efficiency. The computational complexity usually makes these methods unsuitable for solving complex robotics problems in real-time. An example of the use of Monte Carlo methods in robotics is the particle filter-based SLAM algorithm called FastSLAM [5], which only utilizes a particle filter to estimate the posterior of the robots pose and settles for Kalman filters for estimating the pose of landmarks.

The third set of methods, that have gained increasing interest in the last decade due to the advancement in stochastic optimization and increase in computational resources, is the optimization-based method called variational inference. In variational inference, optimization is used to approximate the distribution, $p(z)$, that we are interested in finding, by another simpler distribution $q(z)$, called the variational distribution. Like analytical solutions, variational inference assumes an approximation model, q, and thereby introduces a bias into the solution. The set of possible models that can be employed in modern variational inference is wide, making the method very flexible for modelling robotics problems. This optimization-based approach also makes the distinction between the model of the real problem, p, and the model used to find an approximate solution, q, very explicit and gives a measure of the applicability of the approximate model, q. Furthermore, the use of an approximate model, q, usually allows this set of methods to be more computationally efficient than Monte Carlo methods. As such, variational inference can be viewed as a compromise between the computational efficiency of the model simplification method and the flexibility of Monte Carlo methods. This makes variational inference especially interesting for robotics applications.

Initial efforts on applying variational inference for robot applications have shown promising results in various problems. In [6] variational inference is used to solve several tasks related to navigation in spatial environments for a single robot. In [7] variational inference is used to learn low-level dynamics as well as meta-dynamics of a system, which is subsequently used to plan actions at multiple temporal resolutions. In a similar fashion, it is also demonstrated in [8] how variational inference can be used to learn both low-level and high-level action policies from demonstrations. In [9], variational inference with a mixture model as the variational distribution is used to find approximate solutions to robot configurations satisfying multiple objectives. Variational inference has also been used in some distributed settings. In [10], they perform centralised training with decentralised execution for cooperative deep multi-agent reinforcement learning, where a variational distribution is used in the approximation of a shared global mutual information objective common for all the agents. In [11], variational inference is used to learn a latent variable model that infers the role and index assignments for a set of demonstration trajectories, before these demonstrations are passed to another algorithm that than learns the optimal policy for each agent in a coordinated multi-Agent problem. Common for [10, 11] is that variational inference is used to learn global parameters in a centralized fashion. In [12], a more decentralized approach is taken. Here, variational inference is used locally on each robot in a swarm to estimates a Bayesian Hilbert Map. These locally estimated maps are subsequently merged through a method called Conflation. A method applicable due to an assumption about normal distributed random variables. While others have successfully used variational inference for robotics applications even in distributed settings, the use of a combination of stochastic variational inference and message-passing for decentralized distributed robotic problems has been an untouched topic to date.

In the present effort, we unite these two major solution approaches in variational inference to outline a flexible framework for solving probabilistic robotics problems in a distributed way. The main contribution of this paper is:

- A demonstration of the feasibility of combining stochastic variational inference with message-passing for distributed robotic applications by deriving an algorithm for multi-robot navigation with cooperative avoidance under uncertainty. We validate this through simulations and a real-world experiment with two robots.

In Section B.2, we formally present the basics of variational inference, message-passing, and stochastic variational inference. In Section B.3, we introduce the problem of and derive the algorithm for multi-robot navigation with cooperative avoidance under uncertainty. In Section B.4, we present the results of simulations and a real-world experiment. Finally, in Sections B.5 and B.6, we conclude upon the obtained results and discuss the potential use cases of the proposed approach.

B.2 Variational Inference

Variational inference uses optimization to approximate one distribution $p(z)$ by another, simpler distribution $q(z)$ called the variational distribution. Notice that, in general, $p(z)$ does not need to be a conditional distribution, $p(z|x = \overline{x})$, as in Equation (B.2). However, for the sake of the topic in this paper, we will focus on the conditional distribution case. Thus, we will concentrate on solving a variational inference problem on the form

$$q^{*}(z) = \arg \min_{q(z) \in Q} D\left(p\left(z|x = \overline{x}\right) || q(z)\right), \tag{B.3}$$

where D is a so-called divergence measure, measuring the similarity between p and q, and Q is the family of variational distributions from which we want to find our approximation. The notation $D(x||y)$ denotes that we are dealing with a divergence measure and that the order of arguments, x and y, matters. The family of variational distributions, Q, is usually selected as a compromise between how good an approximation one wants and computational efficiency. The divergence measure, D, can have a rather large impact on the approximation. However, experiments have shown that for the family of α-divergences, subsuming the commonly used Kullback–Leibler divergence, all choices will give similar results as long as the approximating family, Q, is a good fit to the true distribution [13].

Sections B.2.1 and B.2.2 present two solution approaches commonly used in variational inference, namely message-passing algorithms and stochastic variational inference. Message-passing algorithms exploit the dependency structure of a given variational inference problem to decompose the overall problem into a series of simpler variational inference sub-problems, that can be solved in a distributed fashion [13]. Message-passing algorithms do not give specific directions on how to solve these sub-problems, and thus classically required tedious analytical derivations, that effectively limited the usability of the method. On the other hand, modern stochastic variational inference methods directly solve such variational inference problems utilizing stochastic optimization that inherently permits the incorporation of modern machine learning models, such as artificial neural networks, into the problem definition [14, 15]. As such, the fusion of these two approaches can potentially result in a transparent and flexible framework in which complex problems can be solved distributively, making it a perfect fit for a broad interdisciplinary research area such as robotics, inherently accommodating recent trends in research fields such as deep learning, cloud robotics and multi-robot systems.

B.2.1 Message-Passing

The overall idea behind message-passing algorithms is to take a possible complicated problem as defined by Equation (B.3) and break it down into a series of more tractable problems that depend on the solution of the other problems [13, 16]. This way of solving a variational inference problem is known as message-passing because the solution of each sub-problem can be interpreted as a message sent to the other sub-problems. This is achieved by assuming that the model of our problem, $p(z|x)$, naturally factorizes into a product of probability distributions

$$p(z|x) = \prod_{a\in A} p^{(a)}(z|x), \tag{B.4}$$

where superscript (a) is used to denote the index of the a'th factor. Notice that the factorization need not be unique and that each probability distribution, $p^{(a)}(z|x)$, can depend on any number of the variables of $p(z|x)$. The choice is up to us. Similarly, we can choose a variational distribution, $q\left(z\right)$, that factorizes into a similar form

$$q\left(z\right) = \prod_{a\in A} q^{(a)}(z). \tag{B.5}$$

Now by defining the product of all other than the a'th factor of $q\left(z\right)$ and $p(z|x)$, respectively as

$$q^{\setminus a}\left(z\right) = \prod_{b\in A\setminus a} q^{(b)}(z), \tag{B.6}$$

$$p^{\setminus a}\left(z|x\right) = \prod_{b\in A\setminus a} p^{(b)}(z|x), \tag{B.7}$$

and by further assuming that $q^{\setminus a *}\left(z\right) \approx p^{\setminus a}\left(z|x\right)$ is in fact a good approximation, it is possible to rewrite our full problem in Equation (B.3) into a series of approximate sub-problems on the form

$$q^{(a)*}(z) \approx \arg\min_{q^{(a)}\in Q^{(a)}} D\left[p^{(a)}(z|x)q^{\setminus a}\left(z\right)||q^{(a)}(z)q^{\setminus a}\left(z\right)\right]. \tag{B.8}$$

Assuming a sensible choice of factor families, $Q^{(a)}$, from which $q^{(a)}$ can be chosen, the problem in Equation (B.8) can be more tractable than the original problem, and by iterating over these coupled sub-problems as shown in Algorithm B.1, we can obtain an approximate solution to our original problem.

```
1: Initialize $q^{(a)*}(z)$ for all $a \in A$
2: repeat
3:     Pick a factor $a \in A$
4:     Solve Equation (B.8) to find $q^{(a)*}(z)$
5: until $q^{(a)*}(z)$ converges for all $a \in A$
```

Algorithm B.1: The generic message-passing algorithm.

The approach is not guaranteed to converge for general problems. Furthermore, Equation (B.8) might still be a hard problem to solve, thus previously in practice, the approach has been limited to problems for which Equation (B.8) can be solved analytically such as fully discrete or Gaussian problems [13]. However, besides breaking the original problem into a series of more tractable sub-problems, this solution approach also gives a principle way of solving the original problem in a distributed fashion, which can be a huge benefit in robotics applications. Furthermore, depending on the dependency structure of the problem, a sub-problem might only depend on the solution of some of the other sub-problems, which can significantly reduce the amount of communication needed due to sparsely connected networks.

B.2.2 Stochastic Variational Inference

Stochastic Variational Inference (SVI) reformulates the minimization problem of a variational inference problem, e.g., Equation (B.3) or Equation (B.8), into a dual maximization problem with an objective, L, that is suited for stochastic optimization. To use stochastic optimization, we need to assume that the variational distribution, q, is parameterized by some parameters, ϕ. We will denote the parameterized variational distribution by q_ϕ. The steps and assumptions taken to obtain this dual problem and the objective function, L, of the resulting maximization problem of course depends on whether we have chosen the Kullback–Leibler divergence [17–19], α-divergences [20], or another divergence measure [21]. However, the resulting maximization problem ends up being on the form

$$\phi^* = \arg\max_{\phi} \quad \underbrace{L\left(p\left(z, x = \overline{x}\right), q_\phi\left(z\right)\right)}_{E_{z \sim q_\phi(z)}[l(z,\phi)]}. \tag{B.9}$$

This dual objective function, L, does not depend on the posterior, $p\left(z|x = \overline{x}\right)$, but only the variational distribution, $q_\phi\left(z\right)$ and the unconditional distribution $p\left(z, x = \overline{x}\right)$ making the problem much easier to work with. Furthermore, by, for example, utilizing the reparameterization trick or the REINFORCE-gradient, it is possible to obtain an unbiased estimate of the gradient, $\overline{\nabla_\phi L}$, of the dual objective L. Stochastic gradient ascent can then be used to iteratively optimize the objective through the updated equation

$$\phi^{l} = \phi^{l-1} + \rho^{l-1}\overline{\nabla_{\phi} L}^{l}\left(\phi^{l-1}\right), \tag{B.10}$$

where superscript l is used to denote the l'th iteration. If the sequence of learning rates, ρ^{l-1}, follows the Robbins–Monro conditions,

$$\sum_{l=1}^{\infty} \rho^{(l)} = \infty, \qquad \sum_{l=1}^{\infty} \left(\rho^{(l)}\right)^{2} < \infty, \tag{B.11}$$

then stochastic gradient ascent converges to a maximum of the objective function L, and Equation (B.9) is dual to the original minimization problem, thus providing a solution to the original problem.

An unbiased gradient estimator with low variance is pivotal for this method, and variance reduction methods are often necessary. However, a discussion of this subject is outside the scope of this paper and can often be achieved automatically by probabilistic programming libraries/languages such as Pyro [14]. Besides providing the basic algorithms for stochastic variational inference, such modern probabilistic programming languages also provide ways of defining a wide variety of probability distributions and extensions to stochastic variational inference that permits incorporating and learning of parameterized functions, such as neural networks, into the unconditional distribution $p\left(z, x = \overline{x}\right)$, thereby making the approach very versatile. The benefit of solving variational inference problems with stochastic optimization is that noisy estimates of the gradient are often relatively cheap to compute due to, e.g., subsampling of data. Furthermore, the use of noisy gradient estimates can cause algorithms to escape shallow local optima of complex objective functions [19].

To summarize, if we want to distribute a complex inference problem, one potential solution is to first find variational inference sub-problems via the message-passing method, and then use stochastic variational inference to solve these sub-problems. This procedure is illustrated in Figure B.1, and the next section explains the usage of our method for a distributed multi-robot system.

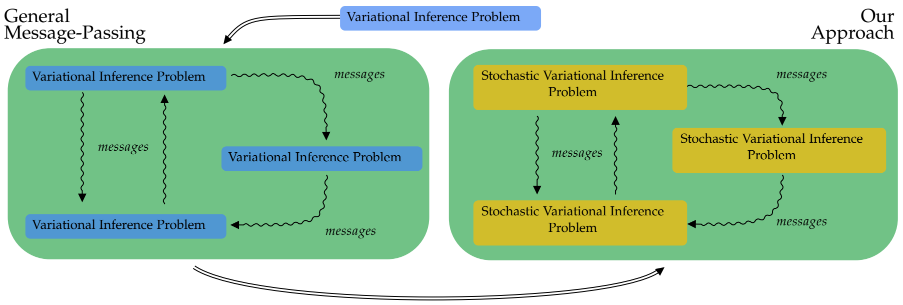

General Message-Passing
Variational Inference Problem
Our Approach
Variational Inference Problem
messages
Variational Inference Problem
messages
Variational Inference Problem
messages
Stochastic Variational Inference Problem
messages
Stochastic Variational Inference Problem
messages
Stochastic Variational Inference Problem
messages

Fig. B.1: We propose to solve complicated robotics problems explicitly, taking uncertainty into account by utilising variational inference as seen in the single blue box. To distribute the necessary computations, we propose to utilise the concept of message-passing algorithms to divide the overall problem into a set of sub-problems that can potentially be sparsely connected, as illustrated in the green box with blue boxes inside. To make these sub-problems computationally tractable, we furthermore propose to solve them utilizing stochastic variational inference as seen in the green box with yellow boxes inside.

B.3 Navigation with Cooperative Avoidance under Uncertainty

Multi-robot collision avoidance is the problem of multiple robots navigating a shared environment to fulfil their respective objective without colliding. It is a problem that arises in many situations such as warehouse management and transportation, collaborative material transfer and construction [22], entertainment [23], search and rescue missions [24], and connected autonomous vehicles [25]. Due to its importance in these and other applications, multi-robot collision avoidance has been extensively studied in the literature. In non-cooperative collision avoidance, each robot assumes that other robots do not actively take actions to avoid collisions, i.e., a worst case scenario. A common approach to non-cooperative collision avoidance is velocity obstacles [26–28]. Velocity obstacles geometrically characterize the set of velocities for the robot that result in a collision at some future time, assuming that the other robots maintains the observed velocity. By only allowing robots to take actions that keep them outside of this set, they avoid collisions. However, non-cooperative approaches are conservative by nature as they neglect the fact that other robots, in most cases, will also try to avoid collisions. Cooperative collision avoidance alleviates this conservatism by assuming that the responsibility of avoiding collisions is shared between the robots. Such approaches include the extensions to velocity obstacles referred to as reciprocal collision avoidance [29–32], but also includes approaches relying on centralized computations of actions, and decentralized approaches in which robots communicate their intentions to each other. For both non- and cooperative collision avoidance, action decision is commonly based on a deterministic optimization/model predictive control formulations [28, 33–35]. However, optimal control [36], Lyapunov theory [37, 38], and even machine learning approaches [39] have also been used.

Despite many claims of guaranteed safety in the literature, uncertainty is often totally neglected, treated in an inapt way, or only to a limited extent. An inapt but common approach to handle uncertainties is to derive deterministic algorithms assuming no uncertainties, and afterwards artificially increase the size of robots used in the algorithm by an arbitrary number, as in [28, 30]. For example, in [30], uncertainties are handled by artificially increasing the radii of robots with 33%. Despite being stated otherwise in the paper, it is clear from the accompanying video material (`https://youtu.be/s9lvMvFcuCE?t=144` Accessed on 23 Marts 2022) that this is not sufficient to avoid contact between robots during a real-world experiment. When uncertainty is treated in an appropriate way, it is usually only examined for a single source of uncertainty, e.g., position estimation error as in [27, 35, 38], presumably due to the difficulties of other methods mentioned in Section B.1, such as deriving analytical solutions or computing solutions in real-time, which is only further complicated by the need for distributed solutions.

Within this section, we illustrate how the approach outlined in Section B.2 can be utilized to solve the multi-robot collision avoidance problem in a cooperative and distributed way that appropriately treats multiple sources of uncertainty. Section B.3.1 introduces the problem dealt with in this paper, in Section B.3.2 the algorithm is derived and explained, and finally, in Section B.4, the result of simulations and a real-world experiment is presented, validating the approach.

B.3.1 Problem Definition and Modelling

Consider N uni-cycle robots placed in the same environment. Each of them have to navigate to a goal location, $z_{g,n} = \left[z_{x,g,n}, z_{y,g,n}\right]^T$, by controlling their translational and rotational velocities while communicating with the other robots to avoid collision. We will consider the two-dimensional case where the robots can obtain a mean and covariance estimate of their own current pose at time t, $z^t_{q,n} = \left[z^t_{x,n}, z^t_{y,n}, z^t_{\psi,n}\right]^T$, e.g., from a standard localization algorithm such as Adaptive Monte Carlo Localization (AMCL) from the Nav2 ROS2 package [40]. Therefore, we model the current pose of the n'th robot as the following normal distribution

$$p(z^t_{q,n}) = N(\mu_{z^t_{q,n}}, \sigma_{z^t_{q,n}}). \tag{B.12}$$

We do not consider the dynamics of the robots but settle for a standard discrete kinematic motion model of a uni-cycle robot given by

$$z_{q,n}^{\tau+1} = \underbrace{z_{q,n}^{\tau} + \begin{bmatrix} \cos\left(z_{\psi,n}^{\tau}\right) & 0 \\ \sin\left(z_{\psi,n}^{\tau}\right) & 0 \\ 0 & 1 \end{bmatrix} A\left(z_{a,n}^{\tau}\right)\Delta T,}_{f\left(z_{q,n}^{\tau}, z_{a,n}^{\tau}\right)} \tag{B.13}$$

where $z_{a,n}^{\tau} = \left[z_{a1,n}^{\tau}, z_{a2,n}^{\tau}\right]^{T}$, $z_{a1,n}^{\tau}$ and $z_{a2,n}^{\tau}$ are the translational and rotational velocities of the n'th robot at time τ normalized to the range $[0,1]$, respectively, A is a linear scaling of the velocity to be in the range $\left[\underline{z_{a,n}^{\tau}}, \overline{z_{a,n}^{\tau}}\right]$ corresponding to the minimum and maximum velocities of the n'th robot, and ΔT is the temporal difference between τ and $\tau+1$. As Equation (B.13), among other things, does not consider the dynamics of the motion, an estimate based on this will yield an error. To model this error, we employ an uniform distribution and define

$$p\left(z_{q,n}^{\tau+1} | z_{q,n}^{\tau}, z_{a,n}^{\tau}\right) = U\left(f\left(z_{q,n}^{\tau}, z_{a,n}^{\tau}\right) - M, f\left(z_{q,n}^{\tau}, z_{a,n}^{\tau}\right) + M\right), \tag{B.14}$$

where M is a constant vector that captures the magnitude of the model error. As Equation (B.13) is obtained through the use of the forward Euler method, M could potentially be obtained as an upper bound by analysing the local truncation error. However, this would probably be too conservative. Instead, we consider M as a tuning parameter. The robots do not naturally have any preference for selecting specific translational and rotational velocities, thus, we also model the prior over the normalized velocities as a uniform distribution. That is

$$p\left(z_{a,n}^{\tau}\right) = U(0,1). \tag{B.15}$$

So far, we have modelled everything we need to describe the uncertainty in the motion of each of the robots. Now, we turn to the problem of modelling optimality and constraints. The only criteria of optimality that we will consider are that the robots grow closer to their respective goal locations, $z_{g,n}$. To do so, we define the following simple reward function

$$r\left(z_{q,n}^{\tau}\right) = \sqrt{\left(z_{g,n} - z_{p,n}^{\tau}\right)^{2}}, \tag{B.16}$$

where

$$z_{p,n}^{\tau} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} z_{q,n}^{\tau}.$$

To include the optimality into the probabilistic model, we use a trick commonly utilized in probabilistic Reinforcement Learning and Control [41]. We start by defining a set of binary optimality variables, $x_{O,n}^{\tau}$, for which $x_{O,n}^{\tau} = 1$ denotes that time step τ is optimal for the n'th robot, and conversely $x_{O,n}^{\tau} = 0$ denotes that time step τ is not optimal. We now define the distribution of this optimality variable at time τ, $x_{O,n}^{\tau}$, conditioned on the pose of the robot at time τ, $z_{q,n}^{\tau}$, as

$$p\left(x_{O,n}^{\tau} | z_{q,n}^{\tau}\right) = Bernoulli\left(e^{-c_1 \cdot r\left(z_{q,n}^{\tau}\right)}\right), \tag{B.17}$$

where c_1 is a tuning constant. Notice that, as $r\left(z_{q,n}^{\tau}\right) \geq 0$, it follows that $e^{-c_1 \cdot r\left(z_{q,n}^{\tau}\right)} \in \left[0, 1\right]$. The intuition behind Equation (B.17) is that the state with the highest reward has the highest probability and states with lower reward have exponentially lower probability.

As stated, the robots should avoid colliding with each other. Therefore, we would like to impose a constraint on the minimum distance, d_{min}, that the n'th and m'tn robots should keep. To do so we define

$$c\left(z_{q,n}^{\tau}, z_{q,m}^{\tau}\right) = \begin{cases} 0 & ; d_{n,m}^{\tau} \leq d_{min} \\ d_{n,m}^{\tau} - d_{min} & ; d_{n,m}^{\tau} > d_{min} \end{cases}, \tag{B.18}$$

where $d_{n,m}^{\tau} = \sqrt{\left(z_{p,n}^{\tau} - z_{p,m}^{\tau}\right)^2}$. Similarly, as we modeled optimality we can now also define binary constraint variables, $x_{C,n,m}^{\tau}$, for which $x_{C,n,m}^{\tau} = 1$ denotes that the minimum distance constraint between the n'th and m'tn robot is violated at time τ, and model the constraint by the distribution given by

$$p\left(x_{C,n,m}^{\tau} | z_{q,n}^{\tau}, z_{q,m}^{\tau}\right) = Bernoulli\left(e^{-c_2 \cdot c\left(z_{q,n}^{\tau}, z_{q,m}^{\tau}\right)}\right), \tag{B.19}$$

where c_2 is a tuning constant. Again, when the distance between two robots becomes larger, it has an exponentially lower probability of violating the distance constraint. With the above variable definitions, we can now formulate a solution to the navigation problem at time t as the following conditional probability distribution

$$p\left(z_{a,1}^{t}, \ldots, z_{a,N}^{t} | X_O^t = 1, X_C^t = 0\right) = \int_{Z^t \setminus \left\{z_{a,1}^{t}, \ldots, z_{a,N}^{t}\right\}} p\left(Z^t | X_O^t = 1, X_C^t = 0\right), \tag{B.20}$$

where

$$
\begin{aligned}
X_O^t &= \left\{X_{O,1}^t, \ldots, X_{O,N}^t\right\}, \\
X_{O,n}^t &= \left\{x_{O,n}^{t+1}, \ldots, x_{O,n}^{k\cdot t}\right\}, \\
Z^t &= \left\{Z_1^t, \ldots, Z_N^t\right\}, \\
Z_n^t &= \left\{z_{q,n}^t, z_{a,n}^t, z_{q,n}^{t+1} \ldots, z_{a,n}^{k\cdot t-1}, z_{q,n}^{k\cdot t}\right\}, \\
X_C^t &= \left\{x_{C,1,2}^{t+1}, \ldots, x_{C,N-1,N}^{t+1}, \ldots, x_{C,1,2}^{k\cdot t}, \ldots, x_{C,N-1,N}^{k\cdot t}\right\}.
\end{aligned}
$$

To capitalize, Equation (B.20) states that we are interested in finding the distribution over the next action, $z_{a,n}^t$, that each robot should take conditioned on that it should be optimal, specified by the "observations" $x_O^t = 1$, and should not result in violation of the constraints, specified by the "observations" $x_C^t = 0$. Furthermore, it states that we can obtain this distribution as the marginal to the conditional distribution on the right-hand side of the equal sign. If we can evaluate this problem efficiently in real-time, it will act as probabilistic model predictive control, taking the next k time-steps into account. However, as discussed in the introduction, solving such a problem is, in general, intractable. Therefore, the next section will derive an approximate solution based on message-passing and Stochastic Variational Inference.

B.3.2 Algorithm Derivation

Instead of solving Equation (B.20), in this section we will show how to find an approximate solution based on variational inference. The derived algorithm is shown in Algorithm B.2. At each time step, t, we want to approximate Equation (B.20) by solving the following problem

$$
\min_{q(Z^t)} D\left[p\left(Z^t | X_O^t = 1, X_C^t = 0\right) || q\left(Z^t\right)\right], \tag{B.21}
$$

while making sure that it is easy to obtain the marginals for the variables of interest, $z_{a,1}^t, \ldots, z_{a,N}^t$, from this approximation. To utilize the idea of message-passing, we need to find a natural factorization of the model of the problem. By applying the definition of conditional probability together with the chain rule, and by considering the dependency structure of the model, the conditional probability distribution on the right-hand side of Equation (B.20) can be rewritten as

$$
p\left(Z^t | X_O^t = 1, X_C^t = 0\right) = \frac{p\left(X_C^t = 0 | Z^t\right)}{p\left(X_C^t = 0\right)} \prod_{n \in [1,N]} p\left(Z_n^t | X_{O,n}^t = 1\right). \tag{B.22}
$$

From Equation (B.22), it is seen that the model naturally factorizes into a fraction related to the constraints and N factors related to the pose, actions, and optimality variables of each of the N robots. Thus, it is natural to choose a variational distribution that factorizes as

$$q\left(Z^t\right) = \prod_{n\in[1,N]} q\left(Z_n^t\right). \tag{B.23}$$

Now considering Equation (B.8) we can distribute the computations by letting the n'th robot solve a problem on the form

$$q^*\left(Z_n^t\right) = \arg\min_{q\left(Z_n^t\right)} D\left[\begin{matrix}\frac{p\left(X_C^t=0|Z^t\right)}{p\left(X_C^t=0\right)} p\left(Z_n^t|X_{O,n}^t = 1\right) \\ \cdot \prod_{m\in[1,N]\setminus n} q\left(Z_m^t\right)\end{matrix} \middle\| \prod_{n\in[1,N]} q\left(Z_n^t\right)\right], \tag{B.24}$$

and broadcast the result, $q^*\left(Z_n^t\right)$, to the rest of the vehicles. This could be repeated until convergence, or simply until a solution for the next time step, $t+1$, has to be found. However, Equation (B.24) still includes the unknown term $p\left(X_C^t = 0\right)$. To overcome this hurdle, we utilize stochastic variational inference, for which we can work with the unconditional distribution given by Equation (B.25) instead.

$$\begin{aligned}&\tilde{p}^{(n)}\left(Z^t, X_{O,n}^t = 1, X_C^t = 0\right) = \\ &\qquad p\left(X_C^t = 0|Z^t\right) p\left(X_{O,n}^t = 1|Z_n^t\right) p\left(Z_n^t\right) \prod_{m\in[1,N]\setminus n} q\left(Z_m^t\right),\end{aligned} \tag{B.25}$$

where

$$\begin{aligned}p\left(X_C^t = 0|Z^t\right) &= \prod_{\tau=t+1}^{kt} \prod_{n=1}^{N-1} \prod_{m=n+1}^{N} p\left(x_{C,n,m}^\tau = 0|z_{q,n}^\tau, z_{q,m}^\tau\right), \\ p\left(X_{O,n}^t = 1|Z_n^t\right) &= \prod_{\tau=t+1}^{kt} p\left(x_{O,n}^\tau = 1|z_{q,n}^\tau\right), \\ p\left(Z_n^t\right) &= p(z_{q,n}^t) \prod_{\tau=t}^{kt-1} p\left(z_{q,n}^{\tau+1}|z_{q,n}^\tau, z_{a,n}^\tau\right) p(z_{a,n}^\tau).\end{aligned} \tag{B.26}$$

```
1:  On each of the $n$ robots
2:  repeat
3:      $t \leftarrow t+1$
4:      Get $\mu_{z^t_{q,n}}, \sigma_{z^t_{q,n}}$ from localization algorithm
5:      Initialize $\phi^{t,*}_n = \{\alpha^t_n, \beta^t_n, \dots, \alpha^{kt-1}_n, \beta^{kt-1}_n\}$
6:      repeat
7:          if messages available for $m \in [1, N]\backslash n$ then
8:              Store $\mu_{z^t_{q,m}}, \sigma_{z^t_{q,m}}$ and $\phi^{t,*}_m$
9:          end if
10:         Solve Equation (B.29) to find $\phi^{t,*}_n$
11:         Broadcast $\mu_{z^t_{q,n}}, \sigma_{z^t_{q,n}}$ and $\phi^{t,*}_n$
12:     until $\phi^{t,*}_n$ converges or time is up.
13: until Suitable stop criteria; e.g., goal reached.
```

Algorithm B.2: Navigation with Cooperative Avoidance under Uncertainty

All terms in Equation (B.25) except for the variational distribution, $q\left(Z^t_m\right)$, were defined in Section B.3.1. To choose an appropriate variational distribution, $q\left(Z^t_m\right)$, consider Equation (B.26) describing the motion of the robot. The only distribution in Equation (B.26) that can actually be directly controlled is $p(z^\tau_{a,n})$, as $p(z^t_{q,n})$ is the current best estimate of the n'th robots' current location provided by a localization algorithm, and $p\left(z^{\tau+1}_{q,n}|z^\tau_{q,n}, z^\tau_{a,n}\right)$ is derived from the kinematics of the robots. Therefore, an appropriate choice of variational distribution is

$$q\left(Z^t_n\right) = p(z^t_{q,n}) \prod_{\tau=t}^{kt-1} p\left(z^{\tau+1}_{q,n}|z^\tau_{q,n}, z^\tau_{a,n}\right) q(z^\tau_{a,n}), \tag{B.27}$$

leaving only the distribution $q(z^\tau_{a,n})$ left to be chosen. $q(z^\tau_{a,n})$ has a direct connection to $p\left(z^t_{a,n}\right)$ in Equation (B.15), and thus it is natural to choose a distribution that shares some of the same properties such as the support. Therefore, we have chosen

$$q(z^\tau_{a,n}) = Beta\left(\alpha^\tau_n, \beta^\tau_n\right), \tag{B.28}$$

which has the exact same support as and even subsumes $p\left(z^t_{a,n}\right)$. To summarize, at each time-step, t, each robot, n, has to iteratively solve a sub-problem through stochastic variational inference represented by

$$\arg\max_{\phi^t_n} \; L\left(\tilde{p}^{(n)}\left(Z^t, X^t_{O,n} = 1, X^t_C = 0\right), q\left(Z^t\right)\right), \tag{B.29}$$

where $\phi^t_n = \{\alpha^t_n, \beta^t_n, \dots, \alpha^{kt-1}_n, \beta^{kt-1}_n\}$, and broadcast the result $\phi^{t,*}_n$ to the other vehicles as illustrated in Figure B.2. In practice, to ease the computational burden, some of the terms can be removed from Equation (B.29), as only the evaluation of the constraints involving the n'th robot is non-constant. Overall we have divided the original approximation problem in Equation (B.21) into a series of less computationally demanding sub-problems that can be solved distributively by each of the robots. The next section presents a simulation study and a real world experiment utilizing this algorithm to make multiple robots safely navigate the same environment, and we will refer to it as "Stochastic Variational Message-passing for Multi-robot Navigation" (SVMMN).

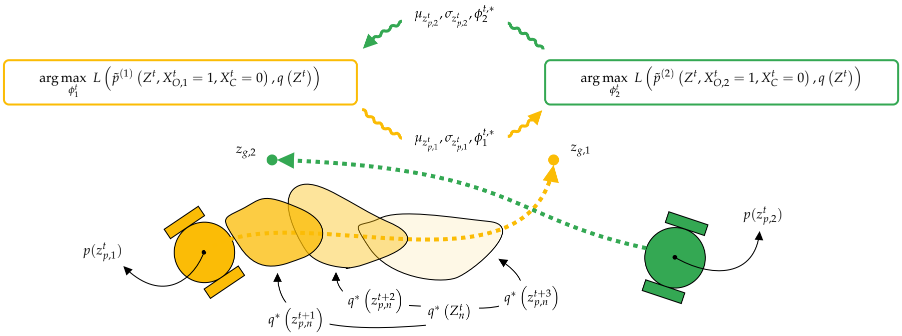

$\mu_{z^t_{p,2}}, \sigma_{z^t_{p,2}}, \phi_2^{t,*}$
$\arg\max_{\phi_1^t} L\left(\tilde{p}^{(1)}\left(Z^t, X^t_{O,1} = 1, X^t_C = 0\right), q\left(Z^t\right)\right)$
$\arg\max_{\phi_2^t} L\left(\tilde{p}^{(2)}\left(Z^t, X^t_{O,2} = 1, X^t_C = 0\right), q\left(Z^t\right)\right)$
$\mu_{z^t_{p,1}}, \sigma_{z^t_{p,1}}, \phi_1^{t,*}$
$z_{g,2}$
$z_{g,1}$
$p(z^t_{p,2})$
$p(z^t_{p,1})$
$q^*\left(z^{t+2}_{p,n}\right)$
$q^*\left(Z^t_n\right)$
$q^*\left(z^{t+3}_{p,n}\right)$
$q^*\left(z^{t+1}_{p,n}\right)$

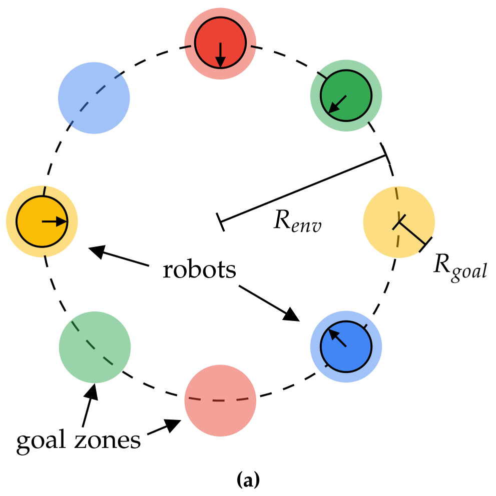
Fig. B.2: The derived algorithm for cooperative navigation under uncertainty of multiple uni-cycle type robots works by letting each robot solve a sub-problem with stochastic variational inference and broadcast the solution to the other robots. Based on the broadcasted solution, a robot implicitly derives a distribution over the other robot's future positions, $q^*\left(z^\tau_{p,n}\right); \tau > t$, and uses the information in its sub-problem.

B.4 Validation

To validate SVMMN described in Section B.3, we performed both a simulation study and a real-world experiment, described in Sections B.4.1 and B.4.2, respectively. In both cases, the models described in Section B.3.1 were implemented utilizing the probabilistic programming language Pyro [14], and Pyro's build-in stochastic variational inference solver was used. For solver options, we chose "Trace_ELBO", thus implicitly using the Kullback–Leibler divergence, and the commonly used "Adam" stochastic optimization solver with 10 epochs/iterations per sent message.

B.4.1 Simulations

To evaluate the stochastic properties of SVMMN, we implemented a simple simulation environment for simulating N uni-cycle robots in parallel and in an asynchronous fashion. For ease of future comparison of algorithms, the environment was created according to the OpenAI Gym API [42]. Currently, the environment implements multiple scenarios commonly used to evaluate multi-robot collision avoidance algorithms: the antipodal goal circle scenario with both evenly distributed initialisation and with random initialisation used in, e.g., [29, 32, 34, 37], and the antipodal circle swapping scenario used in, e.g., [27–35, 38, 39]. Both scenarios are illustrated in Figure B.3. For each of the robots in these scenarios, two goal zones with a radius, R_{goal}, are generated evenly on the circumference of a circle with radius, R_{env}. When the simulation starts, each of the robots are initialized with a position in the centre of one of their respective goal zones. The goal of the robots is then to drive either one time or as many times as possible between the two goal zones without colliding with the robots during the simulated time. While both scenarios cause unnaturally crowded environments, the antipodal circle swapping scenario is specifically designed to provoke reciprocal dances [43] and deadlocks, whereas running the antipodal goal circle scenario with random initialisation for longer durations seems to cause more natural and diverse interactions. Results of simulations for both of these scenarios are summarized in the following sections. The environment, together with the code, data, and an accompanying video for each of the simulations, are available at [44]. Table B.1 in Appendix B summarizes the parameters chosen for the model, environment, and simulations. During the simulations, the real-time factor was adjusted to allow the robots to send approximately 3–4 messages per time-step, imitating the capabilities of the hardware used in the real world experiment.



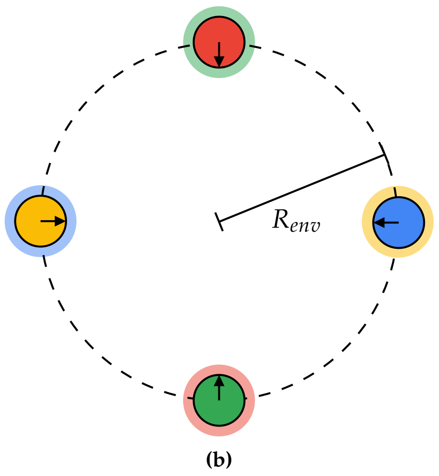


Fig. B.3: Illustration of the simulated environment with $N = 4$. (**a**) Shows the antipodal goal circle scenario and (**b**) shows the antipodal circle swapping scenario. Two goal zones are generated for each of the robots, and the robot is initialized in the center of one of these goal zones.

The Antipodal Goal Circle Scenario

A series of 50 simulations of the antipodal goal circle scenario with 12 robots, and a simulated duration of 300 s were conducted. The simulations were performed with robots having physical properties similar to the robots used in the real world experiment. To quantify the ability of SVMMN to avoid collisions, we utilize the minimum separating distance (MSD) metric also used in [45]. We calculate the MSD of the i'th simulation as the minimum distance between any of the robots during the whole simulation:

$$\text{MSD(i)} = \min_{t \in [0,300]} SSD(t,i),$$

where

$$\begin{aligned} SSD(t,i) &= \min_{\substack{n \in [1, N-1] \\ m \in [n+1, N]}} Dist\left(S_n^{t,i}, S_m^{t,i}\right) && \text{(B.30)} \\ &= \min_{\substack{n \in [1, N-1] \\ m \in [n+1, N]}} ||z_{p,n}^{t,i} - z_{p,m}^{t,i}|| - R_n - R_m, && \text{(B.31)} \end{aligned}$$

$S_n^{t,i}$ is the geometrical set describing the shape and position of the n'th robots at time t, $z_{p,n}^{t,i}$ is the position of the n'th robot at time t in the i'th simulation, R_n^i is the radius of the n'th robot, and the last equality in Equation (B.31) assumes that a disk can describe the robots. The MSD metric has the natural interpretation that a value strictly greater than 0 implies no collisions. Figure B.4 shows the smallest separating distance (SSD) between any of the robots during all of the simulation, together with the mean of the MSD of the 50 simulations, and the smallest MSD recorded in any of the simulations. As seen from Figure B.4, the MSD is strictly greater than 0 at any time, thus 0 collisions occurred. Figure B.5 shows how many times the robots reached a goal zone during the simulations. These plots indicate that no deadlock occurred during the simulations. From these simulations, it can be concluded that SVMMN successfully manages to guide the robots towards their goals while still avoiding collisions.

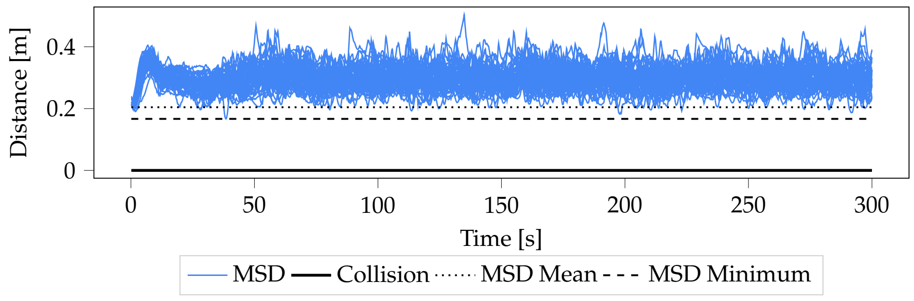


Fig. B.4: The smallest separating distance between any of the 12 robots during each of the 50 simulations.

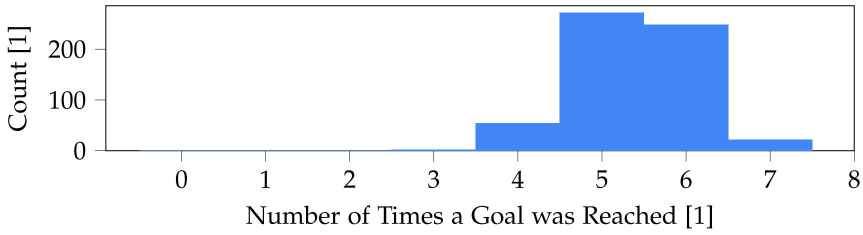


Fig. B.5: Histogram of the number of times the robots reached their goal zones during the simulations.

The Antipodal Circle Swapping Scenario

A series of 10 simulations of the antipodal goal circle scenario with 4, 8, 16, and 32 robots were conducted. The simulations were stopped when all robots had reached their first goal. To make the results comparable with the B-UAVC algorithm [35], we adjusted the simulated radius of the robots together with the noise parameters to fit those used for simulations in [35]. Figure B.6 presents the results of the simulation. As we do not use the same maximum velocities as in [35], we have normalized the travel distance and the travel time, with the minimum possible travel distance and time, respectively. As indicated by the MSD in Figure B.6 zero collisions occurred during these simulations as well. Comparing SVMMN to B-UAVC, SVMMN generally seems to take a shorter path. For a small number of robots in the environment, B-UAVC has the smallest travel time despite SVMMN taking the shortest path. Data from the simulations reveal that the maximum translational velocity reference generated by SVMMN during the simulations with two robots was only 85.9% of the possible translational velocity. This is despite there being nothing to avoid most of the time, indicating that our approach generally picks velocities conservatively. This conservatism could be explained by the use of the Kullback–Leibler divergence, which tends to make the variational distribution, $q(z)$, cover a larger part of the true distribution,

$p(z|x)$, rather then the most probable mode [13]. However, for a large number of robots, SVMMN still seems superior. Thus, from these simulations it can be concluded that SVMMN performs as well as, if not better than, B-UAVC specifically made for the problem of multi-robot collision avoidance.

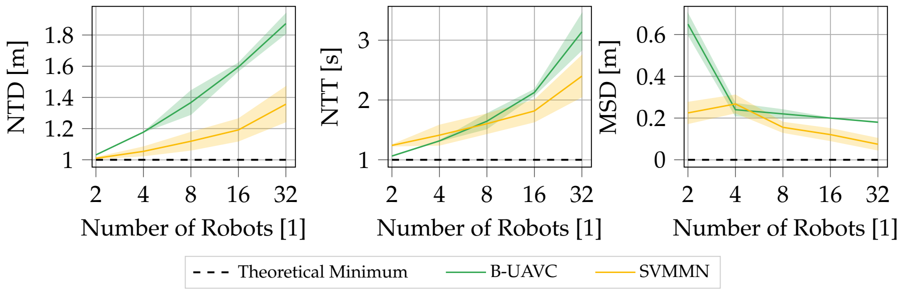


Fig. B.6: Comparison of our algorithm called, "Stochastic Variational Message-passing for Multi-robot Navigation" (SVMMN), based on the combination of message-passing and stochastic variational inference, with B-UAVC [35] made specifically for the problem of multi-robot collision avoidance. The left, middle, and right plot shows the mean and standard deviations of the normalized travel distance (NTD), the normalized travel time (NTT), and the minimum separating distance (MSD) during the simulations, respectively. Notice that the data for B-UAVC are manually obtained from Figure 5 in [35].

B.4.2 Real-Wold Experiment

The real-wold experiment was performed with two TurtleBot3 Burger robots, each equipped with the standard lidar and an Intel NUC10FNK as the on-board processing unit. The parameters chosen for the model are summarized in Table B.1 in Appendix B. To facilitate communication between the robots as needed for message-passing as described in Section B.3.2, the meta operating system ROS2 was utilized. As the communication medium, 5 Ghz Wi-Fi provided by an Asus rt-ax92u router was used. As in the simulations, the robots were programmed to utilize alternating goal locations, $z_{g,n}$, each time they reached within 20cm of their current goal locations. To provide the estimate of the robots current pose distribution, $p(z^t_{q,n})$, we utilized AMCL from the Nav2 ROS2 package [40]. The implemented algorithm is available at [46].

The results of the experiments are shown in Figures B.7–B.9. On average, the robots managed to solve their sub-problem and sent a solution to the other robot 3.01 times per time-step. Figure B.7 shows the full path taken by the robots during the 388 s long experiment. Robot 0 and Robot 1 travelled approximately 37.5 m and 34.8 m, respectively, while reaching their goals 17 times each giving plenty of opportunities for collisions.

Figure B.8 shows the distance between the robots and their respective goal together with the SSD between the two robots themselves during the experiments, and the MSD for the whole test. In this test, the MSD was 7.8 cm. From the plot, it is clear to see that the robots manage to reach their goal several times, while keeping a distance larger than d_{min} from each other.

Figure B.9 illustrates in more detail how SVMMN behaves in one of the situations where the robots were close to each other. To avoid a collision, at time $t = 33$ robot 2 waits for robot 1 to pass. At the time $t = 34$, robot 1 has passed and robot 2 begins planning a trajectory towards its goal and drives towards the goal at $t > 34$. At time $t = 39$, robot 1 has reached one of its goals and starts planning a trajectory towards its other goal. However, for $t > 39$ robot 2 is blocking robots 1's path, and therefore robot 1 does not drive that far.

Overall, the experiment illustrates how SVMMN successfully manages to make the robots drive to their goals while avoiding collisions despite the uncertainty in the localization from AMCL and uncertainty in the future movement of the other vehicle.

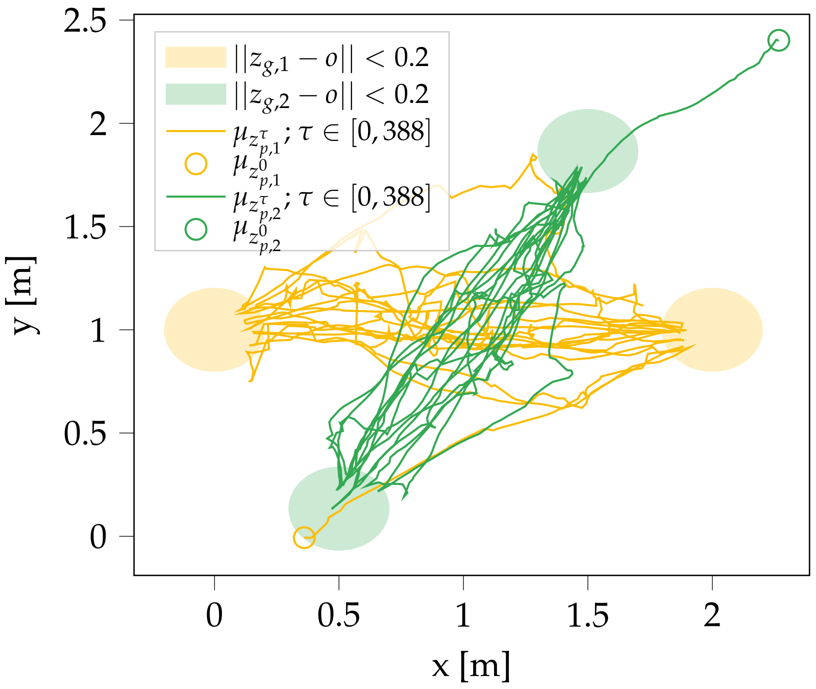


Fig. B.7: Traces of the two robots mean positions, $\mu_{z^t_{p,n}}$, during the experiment, together with the mean of their initial positions distribution, $\mu_{z^0_{p,n}}$, and goal areas defined as a circle with a radius of 20 cm around their goal locations, $z_{g,n}$. The plots show how the robots sometimes deviate from the most direct path between their goal locations to avoid collision with each other.

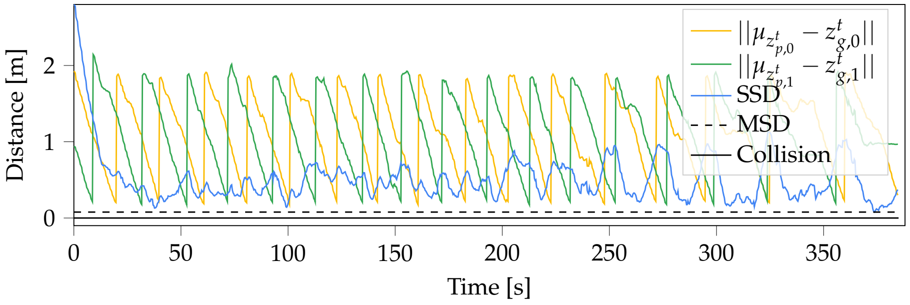


Fig. B.8: Each of the robots' distances to their respective current goals, together with the distance between the two robots for the first 100 s of the experiment, and the minimum distance conservatively calculated as two times the length of the TurtleBot3 Burger platform, 2×138 mm. The jumps in the plots of the robots' distances to their goals are due to change in goal location. The plots show that the robots manage to reach their goal several times without violating a safe distance, d_{min}, from each other.

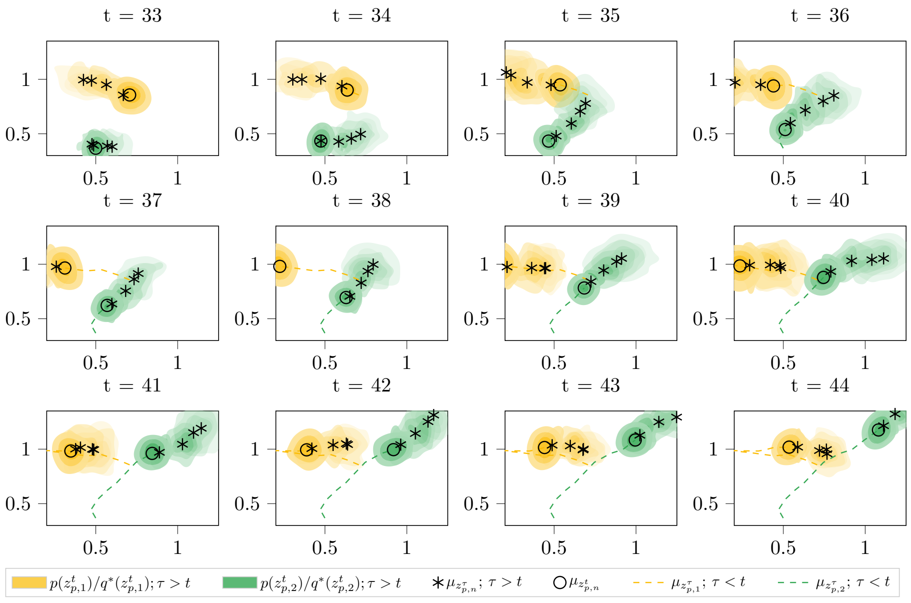


Fig. B.9: Kernel density estimate plots of the robots final predicted positions, $z^\tau_{p,n}; \tau > t$, together with kernel density estimate plot of the robots initial positions, $z^0_{p,n}$, the mean of the samples used to generate each plot, $\mu_{z^\tau_{p,n}}; \tau > t$ and $\mu_{z^\tau_{p,n}}$, and finally the traces of their traversed paths, $\mu_{z^\tau_{p,n}}; \tau < t$, for each $t \in [33, 45]$. The plots clearly illustrate how the robots manage to negotiate trajectories that avoid a collision while taking the relevant uncertainties into account.

B.5 Conclusions

In this paper, we have discussed how variational inference can be a tractable way of solving robotics problems with non-neglectable uncertainties. More specifically, we have shown how two main solution approaches to variational inference, message-passing algorithms, and stochastic variational inference, relate. We outline how these two approaches can be potentially combined to flexibly solve problems with uncertainty in a distributed manner. By deriving and implementing an algorithm for navigation of multiple robots with cooperative avoidance under uncertainty, we furthermore demonstrate the feasibility of the proposed approach. Through simulations, we show that the derived algorithm works with multiple robots, and performs as well as, if not better than, the state of the art algorithm B-UAVC. Finally, we demonstrate that the derived algorithm works in a real-world experiment with two mobile robots.

B.6 Discussion

Many algorithms in robotics are already based on and derived directly from probabilistic models. The wide set of possible models that can be employed in stochastic variational inference should make it straightforward to apply the approach proposed in this paper to many of these probabilistic models, thereby resulting in new and interesting algorithms. Furthermore, due to the separation into sub-problems, the approach could potentially lead the way for offloading more computations to the cloud. As variational inference can incorporate neural networks, the approach also allows for the combination of classical modelling based methods and modern purely learning-based methods.

Author Contributions: Conceptualization, M.D., R.P., and T.B.; Methodology, M.D.; Software, M.D.; Validation, M.D.; Formal Analysis, M.D.; Investigation, M.D.; Writing—Original Draft, M.D.; Writing—Review & Editing, M.D., R.P., and T.B.; Visualization, M.D.; and Supervision, R.P., and T.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The software code used for the simulations and the experiment is available at [44, 46], respectively. Ref. [44] also contains all the data recorded during the simulations together with an .mp4 file animation of each of the simulations.

Conflicts of Interest: The authors declare no conflict of interest.

References

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics.*, ser. Intelligent robotics and autonomous agents. MIT Press, 2005.

[2] C. Zhang, J. Bütepage, H. Kjellström, and S. Mandt, "Advances in variational inference," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 8, pp. 2008–2026, 2019, article in a periodical.

[3] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[4] M. Pfingsthorn and A. Birk, "Simultaneous localization and mapping with multimodal probability distributions," *The International Journal of Robotics Research*, vol. 32, no. 2, pp. 143–171, 2013. [Online]. Available: https://doi.org/10.1177/0278364912461540

[5] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam: A factored solution to the simultaneous localization and mapping problem," in *Eighteenth National Conference on Artificial Intelligence*. USA: American Association for Artificial Intelligence, 2002, p. 593–598.

[6] A. Mirchev, B. Kayalibay, M. Soelch, P. van der Smagt, and J. Bayer, "Approximate bayesian inference in spatial environments," 2019.

[7] K. Fang, Y. Zhu, A. Garg, S. Savarese, and L. Fei-Fei, "Dynamics learning with cascaded variational inference for multi-step manipulation," in *Proceedings of the Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, L. P. Kaelbling, D. Kragic, and K. Sugiura, Eds., vol. 100. PMLR, 30 Oct–01 Nov 2020, pp. 42–52. [Online]. Available: http://proceedings.mlr.press/v100/fang20a.html

[8] T. Shankar and A. Gupta, "Learning robot skills with temporal variational inference," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020,

pp. 8624–8633. [Online]. Available: http://proceedings.mlr.press/v119/shankar20b.html

[9] E. Pignat, T. Lembono, and S. Calinon, "Variational inference with mixture model approximation for applications in robotics," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 3395–3401.

[10] A. Mahajan, T. Rashid, M. Samvelyan, and S. Whiteson, "Maven: Multi-agent variational exploration," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/f816dc0acface7498e10496222e9db10-Paper.pdf

[11] H. M. Le, Y. Yue, P. Carr, and P. Lucey, "Coordinated multi-agent imitation learning," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 1995–2003. [Online]. Available: https://proceedings.mlr.press/v70/le17a.html

[12] W. Zhi, L. Ott, R. Senanayake, and F. Ramos, "Continuous occupancy map fusion with fast bayesian hilbert maps," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 4111–4117.

[13] T. Minka, "Divergence measures and message passing," https://www.microsoft.com/en-us/research/publication/divergence-measures-and-message-passing/, Microsoft, Tech. Rep. MSR-TR-2005-173, Jan. 2005. [Online]. Available: https://www.microsoft.com/en-us/research/publication/divergence-measures-and-message-passing/

[14] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. A. Szerlip, P. Horsfall, and N. D. Goodman, "Pyro: Deep universal probabilistic programming," *J. Mach. Learn. Res.*, vol. 20, pp. 28:1–28:6, 2019, article in a periodical. [Online]. Available: http://jmlr.org/papers/v20/18-403.html

[15] R. G. Krishnan, U. Shalit, and D. Sontag, "Structured inference networks for nonlinear state space models," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, ser. AAAI'17. AAAI Press, 2017, p. 2101–2109.

[16] J. Winn and C. M. Bishop, "Variational message passing," *Journal of Machine Learning Research*, vol. 6, no. 23, pp. 661–694, 2005. [Online]. Available: http://jmlr.org/papers/v6/winn05a.html

[17] R. Ranganath, S. Gerrish, and D. Blei, "Black Box Variational Inference," in *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, S. Kaski and J. Corander, Eds., vol. 33. Reykjavik, Iceland: PMLR, Apr. 2014, pp. 814–822, article in proceedings. [Online]. Available: http://proceedings.mlr.press/v33/ranganath14.html

[18] A. Kucukelbir, D. Tran, R. Ranganath, A. Gelman, and D. M. Blei, "Automatic differentiation variational inference," *Journal of Machine Learning Research*, vol. 18, no. 14, pp. 1–45, 2017, article in a periodical. [Online]. Available: http://jmlr.org/papers/v18/16-107.html

[19] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic variational inference," *Journal of Machine Learning Research*, vol. 14, no. 4, pp. 1303–1347, 2013, article in a periodical. [Online]. Available: http://jmlr.org/papers/v14/hoffman13a.html

[20] Y. Li and R. E. Turner, "Rényi divergence variational inference," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016, pp. 1–9, article in proceedings. [Online]. Available: https://proceedings.neurips.cc/paper/2016/file/7750ca3559e5b8e1f44210283368fc16-Paper.pdf

[21] D. Wang, H. Liu, and Q. Liu, "Variational inference with tail-adaptive f-divergence," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/1cd138d0499a68f4bb72bee04bbec2d7-Paper.pdf

[22] F. Augugliaro, S. Lupashin, M. Hamer, C. Male, M. Hehn, M. W. Mueller, J. S. Willmann, F. Gramazio, M. Kohler, and R. D'Andrea, "The flight assembled architecture installation: Cooperative construction with flying machines," *IEEE Control Systems Magazine*, vol. 34, no. 4, pp. 46–64, 2014.

[23] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Dance of the flying machines: Methods for designing and executing an aerial dance choreography," *IEEE Robotics Automation Magazine*, vol. 20, no. 4, pp. 96–104, 2013.

[24] J. P. Queralta, J. Taipalmaa, B. Can Pullinen, V. K. Sarker, T. Nguyen Gia, H. Tenhunen, M. Gabbouj, J. Raitoharju, and T. Westerlund, "Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision," *IEEE Access*, vol. 8, pp. 191 617–191 643, 2020.

[25] S. W. Loke, “Cooperative automated vehicles: A review of opportunities and challenges in socially intelligent vehicles beyond networking,” *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 4, pp. 509–518, 2019.

[26] P. Fiorini and Z. Shiller, “Motion planning in dynamic environments using velocity obstacles,” *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998. [Online]. Available: https://doi.org/10.1177/027836499801700706

[27] D. Claes, D. Hennes, K. Tuyls, and W. Meeussen, “Collision avoidance under bounded localization uncertainty,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 1192–1198.

[28] J. Alonso-Mora, M. Rufli, R. Siegwart, and P. Beardsley, “Collision avoidance for multiple agents with joint utility maximization,” in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 2833–2838.

[29] J. Snape, J. van den Berg, S. J. Guy, and D. Manocha, “Smooth and collision-free navigation for multiple robots under differential-drive constraints,” in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 4584–4589.

[30] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart, *Optimal Reciprocal Collision Avoidance for Multiple Non-Holonomic Robots*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 203–216. [Online]. Available: https://doi.org/10.1007/978-3-642-32723-0_15

[31] M. Rufli, J. Alonso-Mora, and R. Siegwart, “Reciprocal collision avoidance with motion continuity constraints,” *IEEE Transactions on Robotics*, vol. 29, no. 4, pp. 899–912, 2013.

[32] D. Bareiss and J. van den Berg, “Generalized reciprocal collision avoidance,” *The International Journal of Robotics Research*, vol. 34, no. 12, pp. 1501–1514, 2015. [Online]. Available: https://doi.org/10.1177/0278364915576234

[33] J. Alonso-Mora, P. Beardsley, and R. Siegwart, “Cooperative collision avoidance for nonholonomic robots,” *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 404–420, 2018.

[34] L. Wang, A. D. Ames, and M. Egerstedt, “Safety barrier certificates for collisions-free multirobot systems,” *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

[35] H. Zhu, B. Brito, and J. Alonso-Mora, “Decentralized probabilistic multi-robot collision avoidance using buffered uncertainty-aware voronoi cells,” *Autonomous Robots*, vol. 46, pp. 401–420, 2022, article in a periodical.

[36] G. M. Hoffmann and C. J. Tomlin, "Decentralized cooperative collision avoidance for acceleration constrained vehicles," in *2008 47th IEEE Conference on Decision and Control*, 2008, pp. 4357–4363.

[37] M. Shahriari and M. Biglarbegian, "A novel predictive safety criteria for robust collision avoidance of autonomous robots," *IEEE/ASME Transactions on Mechatronics*, pp. 1–11, 2021.

[38] E. J. Rodríguez-Seda, D. M. Stipanović, and M. W. Spong, "Guaranteed collision avoidance for autonomous systems with acceleration constraints and sensing uncertainties," *J. Optim. Theory Appl.*, vol. 168, no. 3, p. 1014–1038, mar 2016. [Online]. Available: https://doi.org/10.1007/s10957-015-0824-7

[39] K. Sivanathan, B. K. Vinayagam, T. Samak, and C. Samak, "Decentralized motion planning for multi-robot navigation using deep reinforcement learning," in *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)*, 2020, pp. 709–716.

[40] S. Macenski, F. Martin, R. White, and J. Ginés Clavero, "The marathon 2: A navigation system," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[41] S. Levine, "Reinforcement learning and control as probabilistic inference: Tutorial and review," Preprint at arXiv, 2018, article on a preprint server or other repository. [Online]. Available: http://arxiv.org/abs/1805.00909

[42] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[43] J. K. Johnson, "The colliding reciprocal dance problem: A mitigation strategy with application to automotive active safety systems," *CoRR*, vol. abs/1909.09224, 2019. [Online]. Available: http://arxiv.org/abs/1909.09224

[44] M. R. Damgaard, "Multi robot planning simulation," https://github.com/damgaardmr/probMind/tree/d0ba27687b373ff04eb790ef38b21ca8572d8c8a/examples/multiRobotPlanning, 2022.

[45] H. Nishimura, N. Mehr, A. Gaidon, and M. Schwager, "Rat ilqr: A risk auto-tuning controller to optimally account for stochastic model mismatch," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, p. 763–770, Apr 2021. [Online]. Available: http://dx.doi.org/10.1109/LRA.2020.3048660

[46] M. R. Damgaard, “Variational inference navigation,” https://github.com/damgaardmr/VI_Nav/tree/8af532f5e6618d46f3498460af6459e57261fc91, 2021.

Appendix B

Table B.1 summarizes the parameters used during the simulations and the real-world experiment.

Table B.1: Parameters used during the simulations and the real-world experiment.

Parameter	General	
$\underline{z^t_{a,n}}$, $\overline{z^t_{a,n}}$	$[-0.22, -2.84]$, $[0.22, 2.84]$	
M	$[0.05, 0.05, 0.10]$	
ΔT	1 s	
k	4	
d_{min}	$R_n + R_m$	
	Real-world Experiment	
	Robot 1	**Robot 2**
R_n	138 mm	
R_{goal}	0.2 m	
$z_{g,n}$	$[0.00, 0.00]^T$, $[2.00, 1.00]^T$	$[0.50, 0.13]^T$, $[1.50, 1.87]^T$
c_1, c_2	3, 25	
	Antipodal Goal Circle	
N	12	
R_n	138 mm $\pm 20\%$	
R_{env}, R_{goal}	2 m, 0.25 m	
c_1, c_2	3, 5	
$\sigma_{z^t_{q,n}}$	$\text{Diag}\left(\frac{0.05\text{ m}}{3}, \frac{0.05\text{ m}}{3}, \frac{2.5^\circ}{3}\right)^2$	
	Antipodal Circle Swapping	
N	$2, 4, 8, 16, 32$	
R_n	200 mm	
R_{env}, R_{goal}	4 m, 0.1 m	
c_1, c_2	2, 5	
$\sigma_{z^t_{q,n}}$	$\text{Diag}\left(\frac{0.12\text{m}}{3}, \frac{0.12\text{m}}{3}, \frac{2.5^\circ}{3}\right)^2$	

Paper C

A Probabilistic Programming Idiom for Active Knowledge Search

Malte Rørmose Damgaard, Rasmus Pedersen, and Thomas Bak

The paper has been published in the
Proceedings of 2022 International Joint Conference on Neural Networks (IJCNN)
Padua, pp. 1–9, 2022.
`https://www.doi.org/10.1109/IJCNN55064.2022.9892094`

Abstract

In this paper, we derive and implement a probabilistic programming idiom for the problem of acquiring new knowledge about an environment. The idiom is implemented utilizing a modern probabilistic programming language. We demonstrate the utility of this idiom by implementing an algorithm for the specific problem of active mapping and robot exploration. Finally, we evaluate the functionality of the implementation through an extensive simulation study utilizing the HouseExpo dataset.

C.1 Introduction

Making decisions under uncertainty to obtain new knowledge about an environment is a recurring problem within robotics. To efficiently solve this problem, the robot needs to continuously learn about its environment while keeping track of the uncertainty about current knowledge. The decision-making is further complicated if an extrinsic reward signal cannot guide the robot and if predefined constraints should be satisfied. The most well-known and studied problem of this type within robotics is probably active mapping and robot exploration [1]. Most solutions to active mapping and robot exploration heavily exploit the structure of the stored knowledge, i.e., the map, to derive efficient algorithms. E.g. for grid map representations, it is common to apply frontier exploration [2–4]. These methods exploit the property, that it is possible to identify frontiers between the knowledge represented by grid cells in a grid map, that the robot is currently certain about, and the knowledge for which it is uncertain. Actions are chosen to guide the robots towards these frontiers, by which the map is explored. While such approaches exploiting problem-specific properties can result in efficient solutions, they do not easily generalize to other types of knowledge. E.g. because such exploration frontiers cannot easily be defined for other types of knowledge. Other solutions to active mapping and robot exploration take a deep-learning approach, to learn an efficient policy for acquiring new knowledge. E.g., in [5] they feed the current knowledge, again in the form of a grid map, into an artificial neural network and let the output of the network control the actions of the robot. They then train the network with a reward equal to the newly discovered area at each time-step, by which they obtain a policy for exploration. While such an approach can be very efficient at specific tasks, end-to-end learning often limits the generalizability of the solution due to a lack of structural transferability. In many cases, the artificial neural network would have to be re-trained to work for other problems requiring other inputs and outputs. Opposite to the problem-specific approaches already mentioned, the goal of cognitive architectures is to create computational entities with general problem-solving capabilities, that should function across a multitude of tasks. In recent years a community consensus about the overall structure and components of cognitive architectures has begun to emerge, called the Standard Model of the Mind [6]. Especially, the realization of the need for an efficient combination of symbolic and statistical processing is a massive change compared to early research in cognitive architectures. In [7] we presented a generalized framework for developing such cognitive architectures for robotics applications. This was done in an effort to standardize work and promote better cooperation. One of the main ideas of the framework is to develop and identify general and reusable fragments of probabilistic programs, i.e., probabilistic programming idioms, for which inference could be done efficiently utilizing variational inference methods. Inspired by some of the main concepts of the Standard Model of the Mind, the goal of the presented efforts is to develop such a general and reusable probabilistic programming idiom for the problem of making decisions under uncertainty in order to obtain new knowledge about an environment. The main contributions of this paper are:

1. Derivation and implementation of the said probabilistic programming idiom,
2. and validation of the said idiom used in an active mapping and robot exploration context through simulations on a large dataset.

We choose to validate the idiom based on the active mapping and robot exploration problem because it is a well-studied problem with a relatively simple problem formulation for which results are easily interpretable via visual inspection of the robot's trajectory. Still, the problem is sufficiently hard due to the non-convex constraints implied by objects in the environment. Section C.2 presents preliminaries necessary to understand the content of the following sections. In Section C.3 the derivation of the probabilistic programming idiom is presented. In Section C.4 the application of the idiom for the active mapping and robot exploration problem is presented, together with the results of an extensive simulation study. Finally, in Section C.5 we conclude upon the presented work, and hint to future lines of research.

C.2 Preliminaries

Within this paper Z is used to denote latent variables, X is used to denote observed variables, and C is used to denote a collection of both types of variables. We use a superscript in curly brackets to indicate the index of a variable. Specially, for time indexes, we indicate the set of indexes of future variables as $\{t\}^+ = \{t+1, ..., t+\overline{T}\}$. Similarly, we indicate the set of indexes of past variables as $\{t\}^- = \{t-\underline{T}, ..., t\}$. We develop our model primarily for approximate inference with stochastic variational inference. In general, variational inference refer to methods that approximates one conditional distribution, $p(z|x = \overline{x})$ with another unconditional distribution, $q(z)$, through an optimization problem of the form

$$\min_{q(z)\in Q} D[p(z|x = \overline{x})||q(z)]$$

where Q is the family of distributions from which q can be picked, and D is a divergence measure quantifying the difference between p and q. In stochastic variational inference, q is assumed to be parameterised by a set of parameters ϕ, and stochastic gradient ascent is used to solve a tractable dual-problem [8]. To solve this dual-problem, we do not need to know the conditional distribution, $p(z|x = \overline{x})$, but only need to specify the unconditional model, $p(z, x = \overline{x})$, making it a lot easier to work with. However, to use stochastic variational inference we need to ensure that our unconditional model, $p(z, x = \overline{x})$, preserves the differentiability of the dual-problem. Within this paper, we will make use of divergence measures from the family of f-divergences, defined by

$$D_f[p(z)||q(z)] = \int_z q(z) f\left(\frac{p(z)}{q(z)}\right) = E_{q(z)}\left[f\left(\frac{p(z)}{q(z)}\right)\right]$$

where f is an arbitrary convex function [9]. Based on f-divergence we can define the f-information measure as

$$\begin{aligned} I_f[z,y] &= D_f[p(z)p(y)||p(z,y)] \\ &= E_{p(y)}[D_f[p(z)||p(z|y)]]. \end{aligned}$$

The commonly used KL-divergence, D_{KL}, and mutual information is defined by $f(u) = -log(u)$ such that

$$D_{\mathrm{KL}}[p(z)||q(z)] = E_{q(z)}[log(q(z)) - log(p(z))].$$

Similarly, the inverse-KL-divergence, $D_{\overline{KL}}$, and Lautum information, I_L, is defined by $f(u) = u \cdot log(u)$, from which we can obtain the conditional Lautum information measure

$$I_L[z,y|x] = E_{p(y|x)}[D_{\overline{KL}}[p(z|x)||p(z|y,x)]] \tag{C.1}$$

$$= E_{p(y|x)}\begin{bmatrix} log(E_{p(z|x)}[p(y|z,x)]) \\ \quad -E_{p(z|x)}[log(p(y|z,x))] \end{bmatrix}. \tag{C.2}$$

For more information about these measures and their properties, we refer the reader to [9] and [10]. Within this paper, we will also be using the following

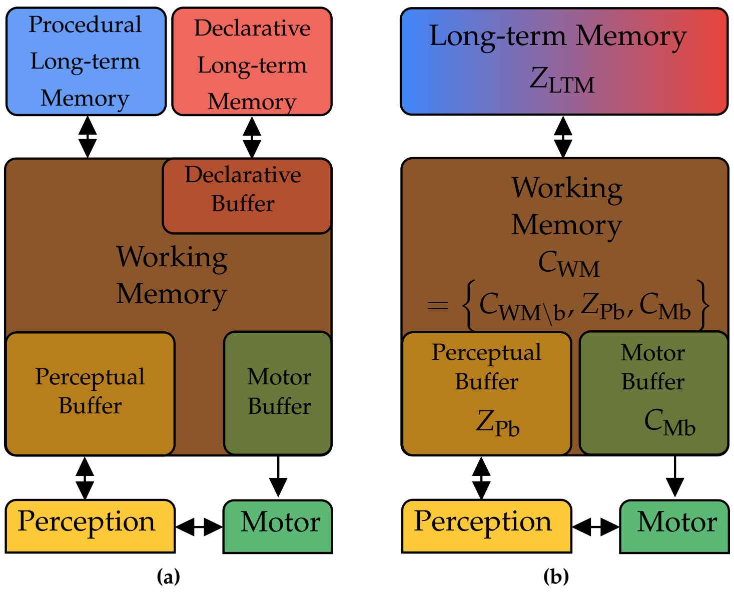

Procedural Long-term Memory
Declarative Long-term Memory
Declarative Buffer
Working Memory
Perceptual Buffer
Motor Buffer
Perception
Motor
(a)
Long-term Memory Z_{LTM}
Working Memory C_{WM} $= \{C_{\text{WM}\backslash\text{b}}, Z_{\text{Pb}}, C_{\text{Mb}}\}$
Perceptual Buffer Z_{Pb}
Motor Buffer C_{Mb}
Perception
Motor
(b)

Fig. C.1: (a) The conceptual memory structure of the Standard Model [6]. The red, blue, brown, green, and yellow colourse are related to declarative long-term memory, procedural long-term memory, working memory, perception component, and motor component, respectively. Here we have used rectangles with rounded corners to symbolise pure memory, and sharp corners to indicate a relation to external signals. (b) The conceptual memory structure used within this paper is without the distinction between procedural and declarative long-term memory, and without the *declarative buffer*. The figure also indicates the symbols used for each type of memory.

approximate ”probabilistic logic”

$$p(z \in \bar{z} \vee y \in \bar{y}) \overset{\text{def}}{=} p(z \in \bar{z}) + p(y \in \bar{y}) - p(z \in \bar{z})p(y \in \bar{y})$$
$$p(z \in \bar{z} \wedge y \in \bar{y}) \overset{\text{def}}{=} p(z \in \bar{z}) \cdot p(y \in \bar{y})$$
$$p\left(\bigwedge_{i=1}^{I} z^{\{i\}} \in \bar{z}^{\{i\}}\right) \overset{\text{def}}{=} \prod_{i=1}^{I} p\left(z^{\{i\}} \in \bar{z}^{\{i\}}\right)$$

where we have used $\vee$ and $\wedge$ to denote the approximate *or* and the *and* operation, respectively. These approximate ”probabilistic logic” rules simply constitute a probabilistic union and intersection with an implied independence assumption, respectively.

C.3 Decision model

According to [6] it is commonly agreed that the memory structure of mind like architectures at a top-level conceptually can be divided into *working memory* and *long-term memory* each of which constitutes relations over symbols supplemented by quantitative metadata to provide a hybrid symbolic-subsymbolic representation. Besides the two main types of memory, it is also agreed that there exists an architectural component denoted *perception* for converting external signals into appropriate memory representations. Similarly, there exists an architectural component denoted *motor* for translating internal memory representations into external signals. The relations between each of the aforementioned are illustrated in Fig. C.1a. *Long-term memory* is responsible for the storage of information over extended periods. The *working memory* includes temporary information necessary for behavior production and problem-solving, such as information about goals, but also contains different buffers for temporarily storing information from the *perception* component, the *motor* component, and some types of long-term memory. As such *working memory* acts as a linkage between the other components. It is customary to sub-divide long-term memory further into specialized types of memories. However, since we in this paper are focusing on decision making to acquire new knowledge, that is updating all types of long-term memory, we will not make such distinctions, as illustrated in Fig. C.1b. Neither will we make a distinction between the declarative buffer and general long-term memory, and jointly refer to them as long-term memory. Furthermore, to keep our presentation relatively concise, we will not consider the relation between working memory, and the *perception* and *motor* components. Instead we will assume that appropriate *perception* and *motor* components are present. To accommodate the need for a hybrid symbolic-subsymbolic representation as suggested by the standard model, we will derive a probabilistic model of decision making. Based on the division of memory and the symbol definitions indicated in Fig. C.1b we make the following definition of the joint probability distribution

$$\begin{aligned} &p\left(C_{\mathrm{WM}\setminus\mathrm{b}}, C_{\mathrm{Mb}}, Z_{\mathrm{Pb}}, Z_{\mathrm{LTM}}\right) \\ &\quad = p\left(Z_{\mathrm{WM}\setminus\mathrm{b}}^{\{t\}^-}, C_{\mathrm{WM}\setminus\mathrm{b}}^{\{t\}^+}, X_{\mathrm{Mb}}^{\{t-1\}^-}, Z_{\mathrm{Mb}}^{\{t-1\}^+}, Z_{\mathrm{Pb}}^{\{t\}^-}, Z_{\mathrm{Pb}}^{\{t\}^+}, Z_{\mathrm{LTM}}\right) \end{aligned}$$

$$\overset{\text{def}}{=} \underbrace{p\left(C_{\text{WM}\backslash\text{b}}^{\{t\}^+}, Z_{\text{Mb}}^{\{t-1\}^+}, Z_{\text{Pb}}^{\{t\}^+} | \breve{Z}_{\text{WM}\backslash\text{b}}^{\{t\}^-}, \breve{Z}_{\text{LTM}}\right)}_{\text{Planning/Decision Making}} \tag{C.3}$$

$$\cdot \underbrace{p\left(Z_{\text{WM}\backslash\text{b}}^{\{t\}^-}, X_{\text{Mb}}^{\{t-1\}^-}, Z_{\text{Pb}}^{\{t\}^-}, Z_{\text{LTM}}\right)}_{\text{Learning/Reasoning}}$$

where we have used sub-script "WM\b" to denote the set of variables representing the working memory except of the set of variables representing the two buffers, i.e., C_{Mb} and Z_{Pb}. In Eq. (C.3) we have assumed that the distribution of future variables, $C_{\text{WM}\backslash\text{b}}^{\{t\}^+}$, $Z_{\text{Mb}}^{\{t-1\}^+}$, and $Z_{\text{Pb}}^{\{t\}^+}$ are conditional independent of previous information in the perceptual buffer, $Z_{\text{Pb}}^{\{t\}^-}$, and motor buffer, $X_{\text{Mb}}^{\{t-1\}^-}$, given the previous variables in the rest of the working memory, $Z_{\text{WM}\backslash\text{b}}^{\{t\}^-}$, and the long-term memory, Z_{LTM}. The last fraction of Eq. (C.3) deals with inference over variables internal to an agent based on past experience in the form of the variables of the perceptual buffer, $Z_{\text{Pb}}^{\{t\}^-}$, and the motor buffer, $Z_{\text{Mb}}^{\{t-1\}^-}$, related to the past. As such this fraction corresponds to reasoning and *learning*. Similarly, the first factor of Eq. (C.3) only deals with future variables based on what have already been learned from past experiences. Since it is assumed that the working memory includes information necessary for behavior production this fraction is responsible for decision making and *planning* guided by preferences contained in the working memory. By the nature of the problem, the probabilistic causation between *learning* and *planning* should only be one way, from *learning* to *planning*. In other words, we can consider inference over the variables in the *learning* part in isolation, and when performing inference in the *planning* part we should keep the *learning* distribution fixed. To emphasize this, we have used breves over the variables $Z_{\text{WM}\backslash\text{b}}^{\{t\}^-}$ and Z_{LTM} in the first fraction of Eq. (C.3). The proposed model effectively divides the cognitive tasks of an agent into *learning* and *planning*. Assuming that we have access to the *learning* distribution, this allows us to focus the rest of the paper on the *planning* part. For the purpose of decision making, and to make our model resemble the classical Markov decision process, we introduce the following variables as a part of the working memory. State variables, Z_{s}, representing the state of the agent itself and the environment. Decision variables, C_{D}, explicitly represent preferences such as goals and constraints. That is, $Z_{\text{s}}^{\{t\}^-} \in C_{\text{WM}\backslash\text{b}}^{\{t\}^-}$ and $\{Z_{\text{s}}^{\{t\}^+}, C_{\text{D}}^{\{t\}^+}\} \in C_{\text{WM}\backslash\text{b}}^{\{t\}^+}$. Furthermore, adopting the Markov property between state variables also used in the Markov decision process we define the planning distribution from Eq. (C.3) as

$$
\begin{aligned}
p & \left(C_{\mathrm{WM}\backslash\mathrm{b}}^{\{t\}^+}, Z_{\mathrm{Mb}}^{\{t-1\}^+}, Z_{\mathrm{Pb}}^{\{t\}^+} | \breve{Z}_{\mathrm{WM}\backslash\mathrm{b}}^{\{t\}^-}, \breve{Z}_{\mathrm{LTM}} \right) \\
& \overset{def}{=} \prod_{\tau=t+2}^{t+\overline{T}} \left[\begin{array}{l} p\left(C_{\mathrm{D}}^{\{\tau\}} | Z_{\mathrm{s}}^{\{\tau\}}, \breve{Z}_{\mathrm{WM}\backslash\mathrm{b}}^{\{t\}^-}, \breve{Z}_{\mathrm{LTM}} \right) \\ \quad \cdot p\left(Z_{\mathrm{s}}^{\{\tau\}} | Z_{\mathrm{s}}^{\{\tau-1\}}, Z_{\mathrm{Mb}}^{\{\tau-1\}} \right) p\left(Z_{\mathrm{Mb}}^{\{\tau-1\}} \right) \end{array} \right] \\
& \quad \cdot p\left(C_{\mathrm{D}}^{\{t+1\}} | Z_{\mathrm{s}}^{\{t+1\}}, \breve{Z}_{\mathrm{WM}\backslash\mathrm{b}}^{\{t\}^-}, \breve{Z}_{\mathrm{LTM}} \right) \\
& \quad \cdot p\left(Z_{\mathrm{s}}^{\{t+1\}} | \breve{Z}_{\mathrm{s}}^{\{t\}}, Z_{\mathrm{Mb}}^{\{t\}} \right) p\left(Z_{\mathrm{Mb}}^{\{t\}} \right).
\end{aligned}
\tag{C.4}
$$

The causality structure of Eq. (C.4) goes as follows. The current possible content of the motor buffer, $Z_{\mathrm{MB}}^{\{\tau\}}$, together with the belief over the state at that time instance, $Z_{\mathrm{s}}^{\{\tau\}}$, determines the belief over the next possible states, $Z_{\mathrm{s}}^{\{\tau+1\}}$. The next possible states, $Z_{\mathrm{s}}^{\{\tau+1\}}$, together with the variables in the long-term memory, Z_{LTM}, and all variables related to the past in the working memory except the buffers, $Z_{\mathrm{WM}\backslash\mathrm{b}}^{\{t\}^-}$, potentially contributes to the current belief over the decision variables, $Z_{\mathrm{D}}^{\{\tau\}}$. Except for the decision variables and the explicit inclusion of the long-term memory variables, most parts of Eq. (C.4) resembles elements known from other decision models such as the Partially observable Markov decision process. As stated earlier, the decision variables are meant to guide the decision process, and as such might be problem-dependent. For the purpose of making decisions to obtain new knowledge, and inspired by [11] we choose to include and combine the following general-purpose decision variables: progress, z_{p}, information gain, z_{i}, constraint, z_{c}, and attention, x_{A}. From these we define $C_{\mathrm{D}}^{\{\tau\}} = \{x_{\mathrm{A}}^{\{\tau\}}, z_{\mathrm{p}}^{\{\tau\}}, z_{\mathrm{i}}^{\{\tau\}}, z_{\mathrm{c}}^{\{\tau\}}\}$. The meaning of these variables are described in the following sections. For reference the structure of the combined model is indicated in Fig. C.2.

C.3.1 Progress

The progress variable is meant to quantify how different a given state, $Z_{\mathrm{s}}^{\{\tau\}}$, is from the past states, $Z_{\mathrm{s}}^{\{\tau\}^-}$. To quantify the progress while taking uncertainty into account we can make use of the divergence measures described in Section C.2. However, calculating such divergence measures inside a probabilistic program amounts to a form of nested inference which potentially can cause problems. E.g. when we want to use stochastic variational inference as the main inference algorithm we have to make sure that we can calculate the gradient of the nested inference performed. Here we choose to use the following one-point estimate of the KL-divergence as a measure of progress

$$\begin{aligned}
&D_{\mathrm{KL}}\left[p\left(Z_{\mathrm{s}}^{\{t-l\}}\right)||p\left(Z_{\mathrm{s}}^{\{\tau\}}|Z_{\mathrm{s}}^{\{\tau-1\}},Z_{\mathrm{Mb}}^{\{\tau-1\}}\right)\right]\\
&\qquad= E_{\hat{Z}_{\mathrm{s}}^{\{\tau\}}}\left[log\left(\frac{p\left(Z_{\mathrm{s}}^{\{\tau\}}=\hat{Z}_{\mathrm{s}}^{\{\tau\}}|Z_{\mathrm{s}}^{\{\tau-1\}},Z_{\mathrm{Mb}}^{\{\tau-1\}}\right)}{p\left(Z_{\mathrm{s}}^{\{t-l\}}=\hat{Z}_{\mathrm{s}}^{\{\tau\}}\right)}\right)\right]\\
&\qquad\approx \frac{1}{I}\sum_{i=1}^{I}\begin{pmatrix} log\left(p\left(Z_{\mathrm{s}}^{\{t-l\}}=\hat{Z}_{\mathrm{s}}^{\{\tau\},\{i\}}\right)\right)\\ -log\left(p\left(Z_{\mathrm{s}}^{\{\tau\}}=\hat{Z}_{\mathrm{s}}^{\{\tau\},\{i\}}|Z_{\mathrm{s}}^{\{\tau-1\}},Z_{\mathrm{Mb}}^{\{\tau-1\}}\right)\right)\end{pmatrix}\\
&\qquad\approx ReLu\begin{pmatrix} log\left(p\left(Z_{\mathrm{s}}^{\{t-l\}}=\hat{Z}_{\mathrm{s}}^{\{\tau\}}\right)\right)\\ -log\left(p\left(Z_{\mathrm{s}}^{\{\tau\}}=\hat{Z}_{\mathrm{s}}^{\{\tau\}}|Z_{\mathrm{s}}^{\{\tau-1\}},Z_{\mathrm{Mb}}^{\{\tau-1\}}\right)\right)\end{pmatrix}\\
&\qquad\overset{\mathrm{def}}{=} P^{\{t-l\}}\left(\hat{Z}_{\mathrm{s}}^{\{\tau\}}\right) \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(C.5)}
\end{aligned}$$

where $p\left(Z_{\mathrm{s}}^{\{t-l\}}\right)$ is a marginal of the learning distribution in Eq. (C.3), $\hat{Z}_{\mathrm{s}}^{\{\tau\}} \sim p\left(Z_{\mathrm{s}}^{\{\tau\}}|Z_{\mathrm{s}}^{\{\tau-1\}},Z_{\mathrm{Mb}}^{\{\tau-1\}}\right)$ and $\hat{Z}_{\mathrm{s}}^{\{\tau\},\{j\}} \sim p\left(Z_{\mathrm{s}}^{\{\tau\}}|Z_{\mathrm{s}}^{\{\tau-1\}},Z_{\mathrm{Mb}}^{\{\tau-1\}}\right)$, and we have used the ReLu function in our approximation since $log\left(\frac{p_1}{p_2}\right) \ngeq 0$ in general but $D_{\mathrm{KL}}[p_1||p_2] \geq 0$. The gradient of the log-probability function can be calculated for many commonly used distributions and probabilistic programs composed of these, and thereby also for this approximation. From this approximation we define the distribution over the progress variable for a given state, $Z_{\mathrm{s}}^{\{\tau\}}$, relative to a single of the past states, $Z_{\mathrm{s}}^{\{t-l\}}$, as

$$p\left(z_{\mathrm{p}}^{\{\tau\},\{l\}}|Z_{\mathrm{s}}^{\{\tau\}}=\hat{Z}_{\mathrm{s}}^{\{\tau\}}\right)=Bernoulli\left(\lambda_p^{\{l\}}\cdot\left[1-e^{-\sigma_p\cdot P^{\{t-l\}}\left(\hat{Z}_{\mathrm{s}}^{\{\tau\}}\right)}\right]\right) \qquad \text{(C.6)}$$

where

$$\lambda_p^{\{l\}}=1-\frac{(1-\lambda_{\mathrm{p,min}})(L-1-l)}{L-1} \quad ; \begin{matrix} L>1\\ L\leqslant \underline{T}\end{matrix}$$

is a decay variable used to put more emphasis on the oldest states considered, L is the number of old states considered, and σ_p is simply a scaling parameter. Here we have used a trick commonly utilised in probabilistic Reinforcement Learning and Control [12], where a given reward is converted to a pseudo probability by exponentiation of that reward. Since, $P^{\{t-l\}}\left(\hat{Z}_{\mathrm{s}}^{\{\tau\}}\right) \geq 0$ it follows that $e^{-\sigma_p\cdot P^{\{t-l\}}\left(\hat{Z}_{\mathrm{s}}^{\{\tau\}}\right)} \in [0,1]$ and thus it can be used as a pseudo probability. Eq. (C.6) thus state that a state, $Z_{\mathrm{s}}^{\{\tau\}}$, yielding a higher approximated divergence, $P^{\{t-l\}}\left(\hat{Z}_{\mathrm{s}}^{\{\tau\}}\right)$, has an exponentially higher probability of yielding progress. From Eq. (C.6) we define the total progress as the combined progress relative to all of the last $L \leq t - \underline{T}$ past states

$$p\left(z_{\mathrm{p}}^{\{\tau\}} | Z_{\mathrm{s}}^{\{\tau\}} = \hat{Z}_{\mathrm{s}}^{\{\tau\}}\right) = Bernoulli\left(p\left(\bigwedge_{l=0}^{L-1} z_{\mathrm{p}}^{\{\tau\},\{l\}} = 1 \middle| Z_{\mathrm{s}}^{\{\tau\}} = \hat{Z}_{\mathrm{s}}^{\{\tau\}}\right)\right). \tag{C.7}$$

The approximation in Eq. (C.5) might seem very coarse; however, when used as nested inference inside a stochastic variational inference algorithm, it is evaluated multiple times during inference of the main problem. The effect is thus effectively similar to a mean approximation using many samples.

C.3.2 Information Gain

As the name implies, the information gain variable, z_i, is meant to quantify the amount of information that can potentially be gained from being in a specific state, $Z_{\mathrm{s}}^{\{\tau\}}$, perceiving the environment and thereby obtaining new information through the perceptual buffer. The perceptual buffer might contain information from multiple independent perceptual modalities, which we will denote as $Z_{\mathrm{Pb}}^{\{\tau\},\{j\}}$. Each of these perceptual modalities might only relate to a specific part of the long-term memory which we will denote $Z_{\mathrm{LTM}}^{\{j\}}$. To quantify the expected amount of information obtained by being in a specific state, $Z_s^{\{\tau\}}$, we use the Lautum information in Eq. (C.1). Based on the Lautum information we represent the pseudo probability that the perceptual modality, $Z_{\mathrm{Pb}}^{\{\tau\},\{j\}}$, will yield new knowledge as the distribution

$$p\left(z_{\mathrm{i}}^{\{\tau\},\{j\}} | Z_{\mathrm{s}}^{\{\tau\}} = \hat{Z}_{\mathrm{s}}^{\{\tau\}}\right) = Bernoulli\left(1 - e^{-\sigma_I \cdot I_{\mathrm{L}}\left[Z_{\mathrm{LTM}}^{\{j\}}, Z_{\mathrm{Pb}}^{\{\tau\},\{j\}} | Z_{\mathrm{s}}^{\{\tau\}} = \hat{Z}_{\mathrm{s}}^{\{\tau\}}\right]}\right) \tag{C.8}$$

where σ_I is a scaling parameter. Maximising the information obtained by each of the perceptual modalities might require wildly different changes to the state, $Z_{\mathrm{s}}^{\{\tau\}}$. Therefore, we focus the attention on the modality providing the most information and define

$$p\left(z_{\mathrm{i}}^{\{\tau\}} | Z_{\mathrm{s}}^{\{\tau\}} = \hat{Z}_{\mathrm{s}}^{\{\tau\}}\right) = Bernoulli\left(\underset{j \in [1,J]}{max}\; p\left(z_{\mathrm{i}}^{\{\tau\},\{j\}} = 1 | Z_{\mathrm{s}}^{\{\tau\}} = \hat{Z}_{\mathrm{s}}^{\{\tau\}}\right)\right).$$

Calculating the Lautum information inside a probabilistic program also amounts to nested inference. To make the calculation of Lautum information compatible with the use of stochastic variational inference for the main problem, we

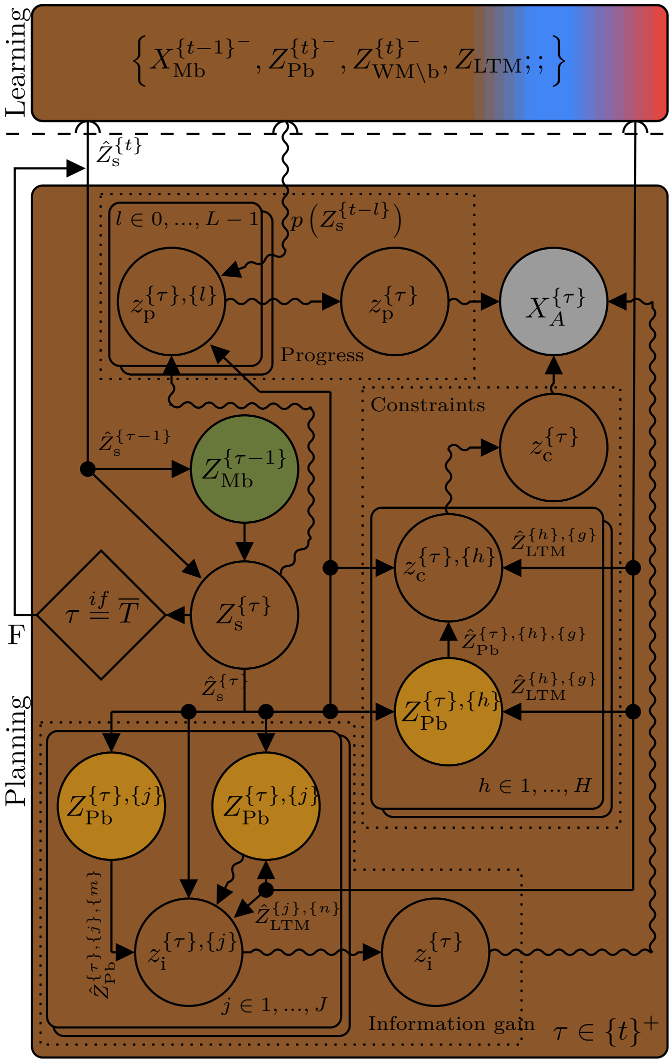

Learning
$\left\{X_{\text{Mb}}^{\{t-1\}^-}, Z_{\text{Pb}}^{\{t\}^-}, Z_{\text{WM}\backslash\text{b}}^{\{t\}^-}, Z_{\text{LTM}};;\right\}$
$\hat{Z}_{\text{s}}^{\{t\}}$
$l \in 0, \ldots, L-1$
$p\left(Z_{\text{s}}^{\{t-l\}}\right)$
$z_{\text{p}}^{\{\tau\},\{l\}}$
$z_{\text{p}}^{\{\tau\}}$
$X_{A}^{\{\tau\}}$
Progress
Constraints
$z_{\text{c}}^{\{\tau\}}$
$\hat{Z}_{\text{s}}^{\{\tau-1\}}$
$Z_{\text{Mb}}^{\{\tau-1\}}$
$z_{\text{c}}^{\{\tau\},\{h\}}$
$\hat{Z}_{\text{LTM}}^{\{h\},\{g\}}$
$\tau \overset{if}{=} T$
F
$Z_{\text{s}}^{\{\tau\}}$
$\hat{Z}_{\text{Pb}}^{\{\tau\},\{h\},\{g\}}$
$\hat{Z}_{\text{s}}^{\{\tau\}}$
$\hat{Z}_{\text{LTM}}^{\{h\},\{g\}}$
$Z_{\text{Pb}}^{\{\tau\},\{h\}}$
Planning
$h \in 1, \ldots, H$
$Z_{\text{Pb}}^{\{\tau\},\{j\}}$
$Z_{\text{Pb}}^{\{\tau\},\{j\}}$
$\hat{Z}_{\text{Pb}}^{\{\tau\},\{j\},\{m\}}$
$\hat{Z}_{\text{LTM}}^{\{j\},\{n\}}$
$z_{\text{i}}^{\{\tau\},\{j\}}$
$z_{\text{i}}^{\{\tau\}}$
$j \in 1, \ldots, J$
Information gain
$\tau \in \{t\}^+$

Fig. C.2: Illustration of the generative flow of the proposed exploration idiom. Rectangles with rounded corners represent a collection of variables. Two stacked rectangles with rounded corners represent conditional independent collections of variables. Circles indicate a distribution over the variable inside the circle. They thereby constitute potential samples sites in the probabilistic program. Solid arrows indicate samples passed around in the probabilistic program, sampled from the distribution represented by the circle at the origin of the arrow. Wavy arrows indicate an evaluation at a specific point of the parent distribution represented by the circle at the origin of the arrow. Gray-colored circles indicate "observed" variables, while other colors are used to indicate a relation to the different types of memories. The rectangles with dotted borders indicate the variables associated with the three different decision variables.

again make use of the following sample mean estimate

$$\begin{aligned}
I_{\mathrm{L}}[x,y|z=\hat{z}] &= E_{\hat{y}}\begin{bmatrix} log(E_{\hat{x}}[p(y=\hat{y}|x=\hat{x},z=\hat{z})]) \\ \quad -E_{\hat{x}}[log(p(y=\hat{y}|x=\hat{x},z=\hat{z}))] \end{bmatrix} \\
&\approx \frac{1}{M}\sum_{m=1}^{M}\begin{bmatrix} log\left(\frac{1}{N}\sum_{n=1}^{N} p\left(y=\hat{y}^{\{m\}}|x=\hat{x},z=\hat{z}\right)\right) \\ -\frac{1}{N}\sum_{n=1}^{N} log\left(p\left(y=\hat{y}^{\{m\}}|x=\ \hat{x}^{\{n\}},z=\hat{z}\right)\right) \end{bmatrix} \\
&= \frac{1}{M}\sum_{m=1}^{M}\begin{bmatrix} log\left(\sum_{n=1}^{N} e^{log\left(p\left(y=\hat{y}^{\{m\}}|x=\ \hat{x}^{\{n\}},z=\hat{z}\right)\right)}\right) \\ -log(N) \\ -\frac{1}{N}\sum_{n=1}^{N} log\left(p\left(y=\hat{y}^{\{m\}}|x=\ \hat{x}^{\{n\}},z=\hat{z}\right)\right) \end{bmatrix}
\end{aligned}$$

where $\hat{x} \sim p(x|z=\hat{z})$ and $\hat{y} \sim p(y|z=\hat{z})$, $\hat{y}^{\{m\}} \sim p(y|z=\hat{z})$ and $\hat{x}^{\{n\}} \sim p(x|z=\hat{z})$. To use the approximation in Eq. (C.8), we only need to be able to evaluate

$$log\left(p\left(Z_{\mathrm{Pb}}^{\{\tau\},\{j\}} = \hat{Z}_{\mathrm{Pb}}^{\{\tau\},\{j\},\{m\}} \,\middle|\, \begin{matrix} Z_{\mathrm{s}}^{\{\tau\}} = \hat{Z}_{\mathrm{s}}^{\{\tau\}}, \\ Z_{\mathrm{LTM}}^{\{j\}} = \hat{Z}_{\mathrm{LTM}}^{\{j\},\{n\}} \end{matrix}\right)\right)$$

with

$$\begin{aligned}
\hat{Z}_{\mathrm{LTM}}^{\{j\},\{n\}} &\sim p\left(Z_{\mathrm{LTM}}^{\{j\}}\right) \\
\hat{Z}_{\mathrm{Pb}}^{\{\tau\},\{j\},\{m\}} &\sim p\left(Z_{\mathrm{Pb}}^{\{\tau\},\{j\}}|Z_{\mathrm{s}}^{\{\tau\}} = \hat{Z}_{\mathrm{s}}^{\{\tau\}}\right)
\end{aligned}$$

where we have assumed that $p\left(Z_{\mathrm{LTM}}^{\{j\}}|Z_{\mathrm{s}}^{\{\tau\}}\right) = p\left(Z_{\mathrm{LTM}}^{\{j\}}\right)$.

C.3.3 Constraints

The constraint variable, $z_c^{\{\tau\}}$, is meant to quantify states, $Z_{\mathrm{s}}^{\{\tau\}}$, that should be avoided taking perceived information, $Z_{\mathrm{PB}}^{\{\tau\}}$, and knowledge stored in long-term memory, Z_{LTM}, into account. Often such constraints can be defined by a set, $A^{\{\tau\},\{h\}}$, that the state, $Z_{\mathrm{s}}^{\{\tau\}}$, should be within. As this set might depend on knowledge stored in the long-term memory, Z_{LTM}, and the expected content of the perceptual buffer, $Z_{\mathrm{Pb}}^{\{\tau\}}$, we assume a set definition of the form

$$A^{\{\tau\},\{h\}} = \left\{Z_{\mathrm{s}}^{\{\tau\}}, Z_{\mathrm{Pb}}^{\{\tau\}}, Z_{\mathrm{LTM}} \mid \mathbb{1}_A^{\{\tau\},\{h\}}\left(Z_{\mathrm{s}}^{\{\tau\}}, Z_{\mathrm{Pb}}^{\{\tau\}}, Z_{\mathrm{LTM}}\right)\right\}$$

where $\mathbb{1}_A^{\{\tau\},\{h\}}$ is the indicator function of the set $A^{\{\tau\},\{h\}}$. Given that $Z_{\mathrm{s}}^{\{\tau\}} = \hat{Z}_{\mathrm{s}}$ the probability that the constraint defined by the set $A^{\{\tau\},\{h\}}$ is satiesfied can then be expressed as

$$P\left(Z_{\text{s}}^{\{\tau\}} = \hat{Z}_{\text{s}}^{\{\tau\}}, Z_{\text{Pb}}^{\{\tau\}}, Z_{\text{LTM}} \in A^{\{\tau\},\{h\}}\right) \\ = E_{\hat{Z}_{\text{Pb}}^{\{\tau\}}, \hat{Z}_{\text{LTM}}}\left[\mathbb{1}_{A}^{\{\tau\},\{h\}}\left(\hat{Z}_{\text{s}}^{\{\tau\}}, \hat{Z}_{\text{Pb}}^{\{\tau\}}, \hat{Z}_{\text{LTM}}\right)\right] \tag{C.9}$$

where $\hat{Z}_{\text{Pb}}^{\{\tau\}} \sim p\left(Z_{\text{Pb}}^{\{\tau\}} | \hat{Z}_{\text{s}}^{\{\tau\}} = \hat{Z}_{\text{s}}^{\{\tau\}}, \hat{Z}_{\text{LTM}} = \hat{Z}_{\text{LTM}}\right)$ and $\hat{Z}_{\text{LTM}} \sim p(Z_{\text{LTM}})$. Based on this we define the distribution over the constraint variable for the h'th constraint at time τ as

$$p\left(z_{\text{c}}^{\{\tau\},\{h\}} | Z_{\text{s}}^{\{\tau\}} = \hat{Z}_{\text{s}}^{\{\tau\}}\right) = \\ \textit{Bernoulli}\left(P\left(Z_{\text{s}}^{\{\tau\}} = \hat{Z}_{\text{s}}^{\{\tau\}}, Z_{\text{Pb}}^{\{\tau\}}, Z_{\text{LTM}} \in A^{\{\tau\},\{h\}}\right)\right)$$

and the distribution over the combined constraint variable at time τ as

$$p\left(z_{\text{c}}^{\{\tau\}} | Z_{\text{s}}^{\{\tau\}} = \hat{Z}_{\text{s}}^{\{\tau\}}\right) = \textit{Bernoulli}\left(p\left(\bigwedge_{h=1}^{H} z_{\text{c}}^{\{\tau\},\{h\}} = 1 \middle| Z_{\text{s}}^{\{\tau\}} = \hat{Z}_{\text{s}}^{\{\tau\}}\right)\right)$$

where H is the number of constraints. Calculating the probability in Eq. (C.9) again amounts to nested inference, but the discontinuity of the indicator function for the set definition, $\mathbb{1}_{A}^{\{\tau\},\{h\}}$, also present a problem for calculating the gradients needed for stochastic variational inference. To overcome this, we assume that the indicator function can be specified as

$$\mathbb{1}_{A}^{\{\tau\},\{h\}}\left(Z_{\text{s}}^{\{\tau\}}, Z_{\text{Pb}}^{\{\tau\}}, Z_{\text{LTM}}\right) = \begin{cases} 1 & d^{\{\tau\},\{h\}}\left(Z_{\text{s}}^{\{\tau\}}, Z_{\text{Pb}}^{\{\tau\}}, Z_{\text{LTM}}\right) > 0 \\ 0 & \textit{else} \end{cases}$$

and make the approximation

$$\mathbb{1}_{A}^{\{\tau\},\{h\}}\left(Z_{\text{s}}^{\{\tau\}}\right) \approx \tilde{\mathbb{1}}_{A}^{\{\tau\},\{h\}}\left(d^{\{\tau\},\{h\}}\left(Z_{\text{s}}^{\{\tau\}}, Z_{\text{Pb}}^{\{\tau\}}, Z_{\text{LTM}}\right)\right)$$

where $\tilde{\mathbb{1}}_{A}^{\{\tau\},\{h\}}(x) \in [0,1]$ is a smooth monotonically increasing function symmetric around $\tilde{\mathbb{1}}_{A}^{\{\tau\},\{h\}}(0) = 0.5$, e.g., a scaled sigmoid function. From this we use the sample mean approximation to obtain the following approximation to the probability in Eq. (C.9)

$$P\left(Z_{\text{s}}^{\{\tau\}} = \hat{Z}_{\text{s}}^{\{\tau\}}, Z_{\text{Pb}}^{\{\tau\}}, Z_{\text{LTM}} \in A^{\{\tau\},\{h\}}\right) \\ \approx \frac{1}{G}\sum_{g=1}^{G} \mathbb{1}_{A}^{\{\tau\},\{h\}}\left(\hat{Z}_{\text{s}}^{\{\tau\}}, \hat{Z}_{\text{Pb}}^{\{\tau\},\{g\}}, \hat{Z}_{\text{LTM}}^{\{g\}}\right) \\ \approx \frac{1}{G}\sum_{g=1}^{G} \tilde{\mathbb{1}}_{A}^{\{\tau\},\{h\}}\left(d^{\{\tau\},\{h\}}\left(\hat{Z}_{\text{s}}^{\{\tau\}}, \hat{Z}_{\text{Pb}}^{\{\tau\},\{g\}}, \hat{Z}_{\text{LTM}}^{\{g\}}\right)\right)$$

where $\hat{Z}_{\text{Pb}}^{\{\tau\},\{g\}} \sim p\left(Z_{\text{Pb}}^{\{\tau\}} | \hat{Z}_{\text{s}}^{\{\tau\}}, \hat{Z}_{\text{LTM}}\right)$ and $\hat{Z}_{\text{LTM}}^{\{g\}} \sim p(Z_{\text{LTM}})$.

C.3.4 Attention

Finally, the attention variable is meant to summarise the other decision variables and symbolises which states the agent should focus its attention on. Based on the approximate "probabilistic logic" presented in Section C.2 we define

$$p\left(x_A^\tau | Z_{\mathrm{s}}^{\{\tau\}} = \hat{Z}_{\mathrm{s}}^{\{\tau\}}\right) = \mathit{Bernoulli}\left(p\left(\left.\begin{array}{l}\left[\begin{array}{l} z_{\mathrm{p}}^{\{\tau\}} = 1 \\ \quad \vee z_{\mathrm{i}}^{\{\tau\}} = 1 \end{array}\right] \\ \qquad\qquad \wedge z_{\mathrm{c}}^{\{\tau\}} = 1 \end{array}\right| Z_{\mathrm{s}}^{\{\tau\}} = \hat{Z}_{\mathrm{s}}^{\{\tau\}}\right)\right). \tag{C.10}$$

Basically, Eq. (C.10) states that an agent should focus its attention on states that either yields progress, *or* yield new knowledge, *and* also satisfies the given constraints.

C.3.5 Variational distribution

As stated in Section C.2, a parameterized unconditional variational distribution, $q\left(Z_{\mathrm{WM}}^{\{t\}^+}\right)$, needs to be specified to utilize stochastic variational inference for approximate inference. Most of the factors in Eq. (C.4) are assumed to be known and thus fixed. Therefore, only the distribution over the variables in the motor buffer can be considered a free distribution, and thus we define

$$\begin{aligned} q\left(Z_{\mathrm{WM}}^{\{t\}^+}\right) &= q\left(Z_{\mathrm{WM}\backslash\mathrm{b}}^{\{t\}^+}, Z_{\mathrm{Mb}}^{\{t-1\}^+}\right) \\ &\overset{def}{=} p\left(Z_{\mathrm{s}}^{\{t+1\}} | \check{Z}_{\mathrm{s}}^{\{t\}}, Z_{\mathrm{Mb}}^{\{t\}}\right) q_{\phi_{\mathrm{Mb}}^{\{t\}}}\left(Z_{\mathrm{Mb}}^{\{t\}} | Z_{\mathrm{s}}^{\{t\}}\right) \\ &\qquad \cdot \prod_{\tau=t+2}^{t+\overline{T}} \left[\begin{array}{l} p\left(Z_{\mathrm{s}}^{\{\tau\}} | Z_{\mathrm{s}}^{\{\tau-1\}}, Z_{\mathrm{Mb}}^{\{\tau-1\}}\right) \\ \quad \cdot q_{\phi_{\mathrm{Mb}}^{\{\tau-1\}}}\left(Z_{\mathrm{Mb}}^{\{\tau-1\}} | Z_{\mathrm{s}}^{\{\tau-1\}}\right) \end{array}\right] \end{aligned}$$

where $\phi_{\mathrm{Mb}}^{\{\tau\}}$ are the parameters that need to be found by stochastic variational inference.

C.3.6 Summery

So far, general functionality that could potentially be utilised for multiple problems has been described and thus could be considered an idiom. This idiom is implemented as an abstract class utilising the probabilistic programming language Pyro [13] developed on top of PyTorch and python. The class contains the following abstract methods that need to be implemented

$q_{\phi_{\mathrm{Mb}}^{\{\tau-1\}}}\left(Z_{\mathrm{Mb}}^{\{\tau-1\}} \vert Z_{\mathrm{s}}^{\{\tau-1\}}\right)$	$p\left(Z_{\mathrm{Pb}}^{\{\tau\},\{j\}} \vert Z_{\mathrm{s}}^{\{\tau\}}, Z_{\mathrm{LTM}}^{\{j\}}\right)$
$p\left(Z_{\mathrm{Mb}}^{\{\tau-1\}} \vert Z_{\mathrm{s}}^{\{\tau-1\}}\right)$	$d^{\{\tau\},\{h\}}\left(Z_{\mathrm{s}}^{\{\tau\}}, Z_{\mathrm{Pb}}^{\{\tau\}}, Z_{\mathrm{LTM}}\right)$
$p\left(Z_{\mathrm{s}}^{\{\tau\}} \vert Z_{\mathrm{s}}^{\{\tau-1\}}, Z_{\mathrm{Mb}}^{\{\tau-1\}}\right)$	$\tilde{\mathbb{1}}_{A}^{\{\tau\},\{h\}}(d)$
$p\left(Z_{\mathrm{Pb}}^{\{\tau\},\{j\}} \vert Z_{\mathrm{s}}^{\{\tau\}}\right)$	$p\left(Z_{\mathrm{LTM}}^{\{j\}}\right)$

The abstract methods representing probability functions need to be implemented as compatible probabilistic programs utilising Pyro. Besides the abstract methods users also need to provide $p\left(Z_{\mathrm{s}}^{\{t\}}\right)$ as a probabilistic program. Besides the above necessary methods, the class also specifies two additional methods that can be used to control the sub-sampling of the long-term memory and perceptual buffer for use in the calculation of information gain and constraint violations. With these methods implemented users can call the class method "*makePlan(...)*" which via stochastic variational inference finds an approximate optimal set of parameters, $q_{\phi_{\mathrm{Mb}}^{\{\tau\},*}}$ to the variational inference problem

$$\min_{\phi_{\mathrm{WM}}^{\{t\}^+}} D\left[\begin{array}{c} p\left(Z_{\mathrm{WM}}^{\{t\}^+} \middle| \begin{array}{c}\breve{Z}_{\mathrm{s}}^{\{t\}}, \breve{Z}_{\mathrm{LTM}}, \\ x_{A}^{\{t\}^+} = 1\end{array}\right) \\ \cdot p\left(Z_{\mathrm{WM}}^{\{t\}^-}, Z_{\mathrm{LTM}}\right)\end{array} \middle\| \begin{array}{c} q_{\phi_{\mathrm{WM}}^{\{t\}^+}}\left(Z_{\mathrm{WM}}^{\{t\}^+} \middle| \begin{array}{c}\breve{Z}_{\mathrm{s}}^{\{t\}}, \\ \breve{Z}_{\mathrm{LTM}}\end{array}\right) \\ \cdot p\left(Z_{\mathrm{WM}}^{\{t\}^-}, Z_{\mathrm{LTM}}\right)\end{array}\right]$$

where the user can specify the divergence measure and optimiser used. The optimal set of parameters is used to draw samples of the future motor buffer

$$Z_{\mathrm{Mb}}^{\{\tau\}} \sim q_{\phi_{\mathrm{Mb}}^{\{\tau\},*}}\left(Z_{\mathrm{Mb}}^{\{\tau\}} \vert Z_{\mathrm{s}}^{\{\tau\}}\right) \qquad ; \tau \in \{t, ..., \overline{T}-1\}.$$

These samples constitute potential future optimal actions needed to optimise information gain or progress while satisfying constraints. Finally, the "*make-Plan(...)*" method either returns these samples or a sample mean hereof. The code is available through [14].

C.4 Autonomous Robot Exploration

To exemplify the utility of the proposed idiom, we have used it to implement an algorithm for autonomous robot exploration. The code for this can be found through [14]. The goal of the implementation is for a robot to explore an environment represented by a grid map autonomously. We consider the problem at a high level, and define the state to be the current position in the

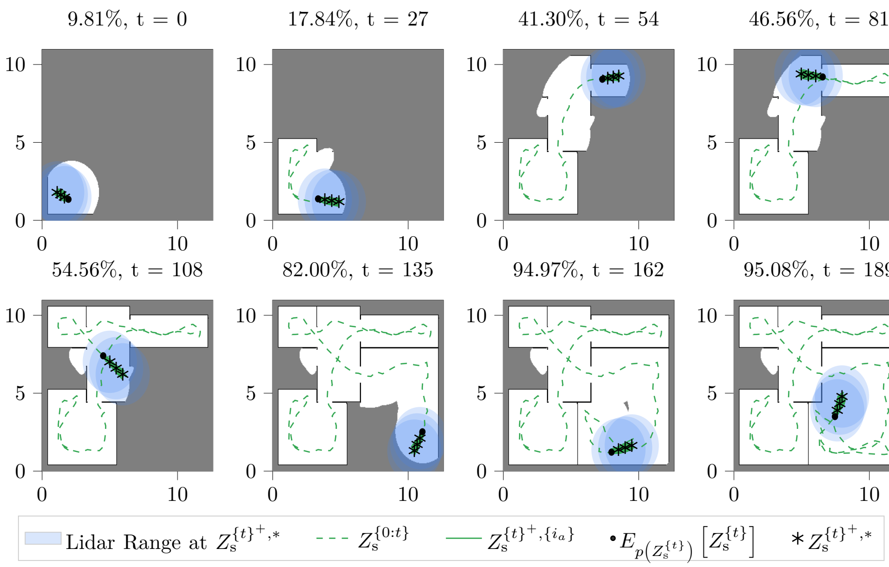


Fig. C.3: The progress of a simulation for the map with ID "7fb9c9203cb8c4404f4af1781f1c6999" after each of the timesteps $t = \{0, 27, 54, 81, 108, 135, 162, 189\}$. For each timestep the previous positions, $Z_s^{\{0:t\}}$, is shown by a green dashed line, the mean of the current position, $Z_s^{\{t\}}$, is shown by a black dot, samples of future positions, $Z_s^{\{\tau\}^+,\{i_a\}}$, is shown by solid green lines, and the mean of these samples corresponding to the optimal future positions based on $Z_a^{\{t\}^+,*}$ are shown by black asterisks. The simulation where terminated after $t = 189$ since the exploration percentage where above 95%.

XY-plane, $Z_s^{\{\tau\}} = \left[z_x^{\{\tau\}}, z_y^{\{\tau\}}\right]^T$, and use the simple transition model as

$$p\left(Z_s^{\{\tau+1\}} | Z_s^{\{\tau\}}, Z_{Mb}^{\{\tau\}}\right) = N\left(Z_s^{\{\tau\}} + A\left(Z_{Mb}^{\{\tau\}}\right), \sigma_a\right)$$

where $Z_{Mb}^{\{\tau\}}$ is the relative position scaled to be in the interval $[0, 1]$, $A(...)$ is a linear scaling of the relative position to be in the range $\left[\underline{\Delta Z_a}, \overline{\Delta Z_a}\right]$, and σ_a is the covariance of the error allowed in the movement. Since the robot should have no prior preference of its movement we define

$$p\left(Z_{Mb}^{\{\tau\}} | Z_s^{\{\tau\}}\right) = U\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}\right). \tag{C.11}$$

$q_{\phi_{Mb}^{\{\tau\}}}\left(Z_{Mb}^{\{\tau\}} | Z_s^{\{\tau\}}\right)$ should have the same support as Eq. (C.11), but should also be flexible enough to represent preferences in the relative position. Thus, we define

$$q_{\phi_{Mb}^{\{\tau\}}}\left(Z_{Mb}^{\{\tau\}} | Z_s^{\{\tau\}}\right) = Beta\left(\alpha_a^{\{\tau\}}, \beta_a^{\{\tau\}}\right)$$

since the beta distribution subsumes the uniform distribution, but also can represent a single mode. Thus, we have $\phi_{\text{Mb}}^{\{\tau\}} = \left\{\alpha_{\text{a}}^{\{\tau\}}, \beta_{\text{a}}^{\{\tau\}}\right\}$. We consider the grid map to be the long-term memory. That is, $Z_{\text{LTM}} = \left\{z_m^{\{1\}}, ..., z_m^{\{I_m\}}\right\}$ where $z_m^{\{i_m\}}$ is each of the cells in the grid map, and make the common assumption that

$$p(Z_{\text{LTM}}) = \prod_{i_m=1}^{I} p\left(z_m^{\{i_m\}}\right)$$

where

$$p\left(z_m^{\{i_m\}}\right) = Bernoulli\left(P_m^{\{i_m\}}\right)$$

and $P_m^{\{i_m\}}$ is the probability of the i_m'th grid cell being occupied. We assume that the environment is perceived through a lidar with 360° field of view and evenly spaced lidar beams with 1° spacing, and define

$$p\left(Z_{\text{Pb}}^{\{\tau\}} | Z_{\text{s}}^{\{\tau\}}, Z_{\text{LTM}}\right) = \prod_{1=i_r}^{360} p\left(z_{\text{r,d}}^{\{\tau\},\{i_r\}} | Z_{\text{s}}^{\{\tau\}}, Z_{\text{LTM}}^{\{\tau\},\{i_r\}}\right)$$

where $z_{\text{r,d}}^{\{\tau\},\{i_r\}}$ is the distance measured by the i_r'th laser beam at time τ given the current position and grid map,

$$Z_{\text{LTM}}^{\{\tau\},\{i_r\}} = \left\{z_m^{\{i_m\}} \in Z_{\text{LTM}} | \text{ray } i_r \text{ intersects cell } i_m\right\}$$

are the cells in the grid map that the i_r'th laser beam intersects. We obtain the set, $Z_{\text{LTM}}^{\{\tau\},\{i_r\}}$, through ray-tracing. The distribution $p\left(z_{\text{r,d}}^{\{\tau\},\{i_r\}} | Z_{\text{s}}^{\{\tau\}}, Z_{\text{LTM}}^{\{\tau\},\{i_r\}}\right)$ is implemented according to the laser beam model in [15]. Without taking the map into consideration the robot have no prior knowledge on the distance measured by the lidar, and thus we define

$$p\left(z_{\text{r,d}}^{\{\tau\},\{i_r\}} | Z_{\text{s}}^{\{\tau\}}\right) = U\left(0, \overline{z_{\text{r,d}}}\right)$$

where $\overline{z_{\text{r,d}}}$ is the max range of the lidar beams. We furthermore want the robot to keep a minimum distance, d_{min}, to occupied cells in the map and thus define the constraints via the logistic function

$$\tilde{\mathbb{1}}_A^{\{\tau\},\{h\}}\left(d^{\{\tau\},\{h\}}\left(Z_{\text{s}}^{\{\tau\}}, Z_{\text{Pb}}^{\{\tau\}}, Z_{\text{LTM}}\right)\right) = \frac{1}{1 + e^{-\sigma_c \cdot \left(z_{\text{r,d}}^{\{\tau\},\{h\}} - d_{min}\right)}}$$

where σ_c determines the steepness of the logistic function. With the above definitions, we have $J = H = 360$. Calculating the information gain and constraint violation based on all 360 lidar beams is computationally intractable in the current implementation. Therefore, for each timestep, τ, we sub-sample the number of lidar beams taken into account by randomly picking $\tilde{J} \ll 360$ and $\tilde{H} \ll 360$ lidar beams for calculating the information gain and constraint violation, respectively. In our implementation, we have furthermore chosen to use Pyro's build-in "ClippedAdam" optimizer with the standard D_{KL} divergence measure. Finally, the next action that the robot should take, $Z_{\mathrm{a}}^{\{t\},*}$, is calculated as the sample mean of optimal actions

$$Z_{\mathrm{a}}^{\{t\},*} = \frac{1}{I_a} \sum_{i_a=1}^{I_a} A\left(Z_{\mathrm{Mb}}^{\{t\},\{i_a\}}\right) \tag{C.12}$$

where $Z_{\mathrm{Mb}}^{\{\tau\},\{i_a\}} \sim q_{\phi_{\mathrm{Mb}}^{\{\tau\},*}}\left(Z_{\mathrm{Mb}}^{\{t\}}|Z_{\mathrm{s}}^{\{\tau\}}\right)$. The calculated $Z_{\mathrm{a}}^{\{t\},*}$ is considered the optimal action for the robot to take in order to maximize progress or the information obtained.

C.4.1 Simulation

To test the algorithm implemented for autonomous robot exploration, we performed simulations on the 35,126 2D floor plans available in the House-Expo dataset utilising a modified version of the accompanying PseudoSLAM simulator [5]. The PseudoSLAM simulator is made to efficiently generate occupancy grid maps directly from 2D floor plans, without the computational burden of running a real SLAM algorithm. The simulator also calculates the percentage of the map that has been explored and keeps a count of the number of crashes. Thereby, the simulator is suitable for large-scale simulation studies. Unfortunately, the original PseudoSLAM simulator only allowed for the three fixed discrete movements: turn θ degrees to the left, turn θ degrees to the right, and move X meters forward, where θ and X are fixed variables. Thus, the original simulator was not suitable for the continuous movements calculated by Eq. (C.12). Therefore, modifications were made to allow for such continuous movements in the simulator. Furthermore, it was found that the function "*measure_ratio()*" build into the PseudoSLAM simulator, meant to quantify the percentage of the map explored, counter-intuitively could return values greater than 1. Thus, we also modified this function. The modified PseudoSLAM simulator is available through [14]. For our simulations, we adopted the simulation procedure used in [5]. One simulation with a random initial position was performed for each of the 35,126 2D floor plans. The simulations were limited to 200 time-steps. They were terminated if the "*measure_ratio()*" function returned more than 0.95, corresponding to more than 95% of the map had been explored. As an example, the result of one of the simulations is illustrated in Fig. C.3. From Fig. C.4 it is seen that for the smallest floor plans in the data set, the robot manages to explore most of its environment. As the size of the floor plans increases, a smaller percentage of the environment is explored on average. This is expected behavior

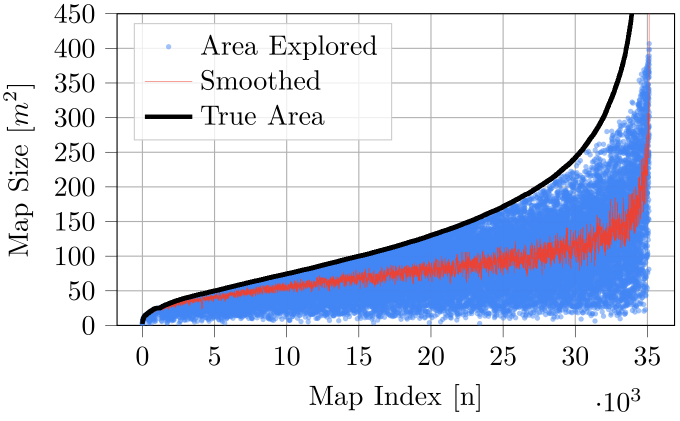


Fig. C.4: The area explored for each of the 35126 simulations performed with the indices sorted in ascending order by the true area of the map. The red curve shows a moving average with a windows size of 20.

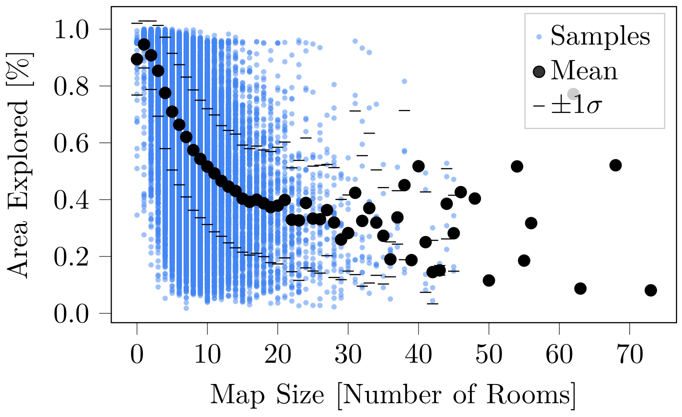


Fig. C.5: The percentage of area explored in each of the 35126 simulations performed compared to the number of rooms in each of the maps.

since there is a limit to how much of a map the robot can explore in a fixed amount of time steps. However, Fig. C.5 might reveal another cause. From Fig. C.5 there seems to be a clear relationship between the number of rooms in the environment, and the percentage of the environment that the robot manages to explore. A possible cause of this could be that for the robot to explore multiple rooms it often has to pass through narrow doorways. Passing through narrow doorways presents a high risk of constraint violation. In many situations, there will be alternative paths away from doorways that still yield progress. Therefore, if the paths going through the doorway does not

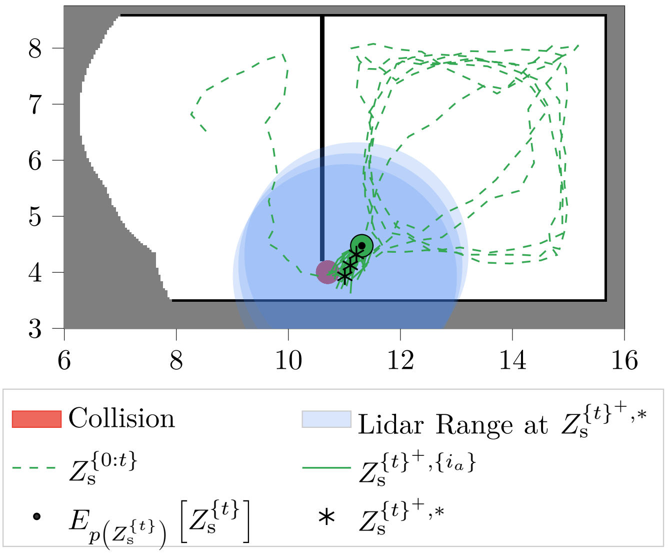


Fig. C.6: The progress of a simulation for the map with ID "23e99dac3228ee2d371c5a627c49e415" after 200 time-steps. In this simulation, the robot spends a lot of time-steps driving around in the same room without getting out of it even though it is fully explored. The figure also shows an example of a collision in a doorway.

yield a high probability of information gain, the presented idiom will prefer actions away from such doorways. This means that the robot could spend more time-steps than necessary in rooms that are fully explored. As an example consider the simulation illustrated in Fig. C.6. In this simulation, the robot starts in "room 1" and passes through a doorway to "room 2" already after a few time-steps. After passing through the doorway, the robot quickly explores the entire "room 2". However, since the area in "room 1" in close vicinity to the doorway is already explored, the probability of information gain for paths passing back through the doorway is low due to the limited lidar range used to define $p\left(Z_{\mathrm{Pb}}^{\{\tau\}} | Z_{\mathrm{s}}^{\{\tau\}}, Z_{\mathrm{LTM}}\right)$. Therefore, the robots keep driving around in "room 2" driven purely by progress. Overcoming this behavior would require some kind of memory about from which of the previous states the robot could obtain more knowledge, and some additional decision variables to guide the robot back to these states. Besides guiding an agent towards new knowledge the idiom is also supposed to avoid constraints. In the implemented robot exploration algorithm, the only constraint is to prevent collisions with the robots surrounding. A total of 1617 unique collisions were recorded in 1253 different maps during the 6469065 time-steps simulated in all of the 35126 2D floor plans. Thus, only $0,25$ ‰ of the time-steps resulted in collisions. Nearly all of these collisions were registered near corners or doorways, like the collision shown in Fig. C.6. Given that the idiom currently only supports checking constraints at discrete states, such behavior is to be expected, since the constraint can be satisfied at two consecutive states but not in between. Furthermore, for the specific application of robot exploration, this small probability of collision would probably be deemed tolerable, since in many cases would have to be a low-level collision avoidance system anyway. If this cannot be tolerated, the idiom would have to modified to include checking of constraint in between the discrete states.

Everything considered the ability of the idiom to guide an agent towards new knowledge while avoiding constraints seems to be as should be expected.

C.5 Discussion

In this paper, we have shown how to develop a generally applicable probabilistic programming idiom for the problem of making decisions under uncertainty to obtain new knowledge about an environment. We based our idiom on the memory structure of the Standard model of mind, and other ideas from research in cognitive architectures. We furthermore show how this idiom can be used for the specific problem of active mapping and robot exploration. Based on an extensive simulation study of this problem, it is concluded that the idiom works as could be expected. The simulation also indicated that the idiom probably would benefit from additional memory of old states in which more knowledge can be obtained. Furthermore, the simulation also indicated that the idiom for some application could benefit from checking constraints in between states.

References

[1] I. Lluvia, E. Lazkano, and A. Ansuategi, "Active mapping and robot exploration: A survey," *Sensors*, vol. 21, no. 7, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/7/2445

[2] A. Topiwala, P. Inani, and A. Kathpal, "Frontier based exploration for autonomous robot," 2018.

[3] E. Uslu, F. Çakmak, M. Balcılar, A. Akıncı, M. F. Amasyalı, and S. Yavuz, "Implementation of frontier-based exploration algorithm for an autonomous robot," in *2015 International Symposium on Innovations in Intelligent SysTems and Applications (INISTA)*, 2015, pp. 1–7.

[4] D. A. Perkasa and J. Santoso, "Improved frontier exploration strategy for active mapping with mobile robot," in *2020 7th International Conference*

on Advance Informatics: Concepts, Theory and Applications (ICAICTA), 2020, pp. 1–6.

[5] T. Li, D. Ho, C. Li, D. Zhu, C. Wang, and M. Q.-H. Meng, "Houseexpo: A large-scale 2d indoor layout dataset for learning-based algorithms on mobile robots," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5839–5846, article in proceedings.

[6] J. E. Laird, C. Lebiere, and P. S. Rosenbloom, "A standard model of the mind: Toward a common computational framework across artificial intelligence, cognitive science, neuroscience, and robotics," *AI Magazine*, vol. 38, no. 4, pp. 13–26, Dec. 2017, article in a periodical. [Online]. Available: https://ojs.aaai.org/index.php/aimagazine/article/view/2744

[7] M. R. Damgaard, R. Pedersen, and T. Bak, "Toward an idiomatic framework for cognitive robotics," 2021.

[8] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic variational inference," *Journal of Machine Learning Research*, vol. 14, no. 4, pp. 1303–1347, 2013, article in a periodical. [Online]. Available: http://jmlr.org/papers/v14/hoffman13a.html

[9] D. Palomar and S. Verdu, "Lautum information," in *2006 IEEE Information Theory Workshop - ITW '06 Punta del Este*, vol. 54, no. 3, 2006, pp. 964 – 975.

[10] T. Minka, "Divergence measures and message passing," https://www.microsoft.com/en-us/research/publication/divergence-measures-and-message-passing/, Microsoft, Tech. Rep. MSR-TR-2005-173, Jan. 2005. [Online]. Available: https://www.microsoft.com/en-us/research/publication/divergence-measures-and-message-passing/

[11] P. S. Rosenbloom, J. Gratch, and V. Ustun, "Towards emotion in sigma: From appraisal to attention," in *Artificial General Intelligence*, J. Bieger, B. Goertzel, and A. Potapov, Eds. Cham: Springer International Publishing, 2015, pp. 142–151. [Online]. Available: https://doi.org/10.1007%2F978-3-319-21365-1_15

[12] S. Levine, "Reinforcement learning and control as probabilistic inference: Tutorial and review," Preprint at arXiv, 2018, article on a preprint server or other repository. [Online]. Available: http://arxiv.org/abs/1805.00909

[13] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. A. Szerlip, P. Horsfall, and N. D. Goodman, "Pyro: Deep universal probabilistic programming," *J. Mach. Learn. Res.*, vol. 20, pp. 28:1–28:6, 2019, article in a periodical. [Online]. Available: http://jmlr.org/papers/v20/18-403.html

[14] M. R. Damgaard, "probmind," Jan. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.5841292

[15] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics.*, ser. Intelligent robotics and autonomous agents. MIT Press, 2005.

Paper D

Escaping Local Minima Via Appraisal Driven Responses

Malte Rørmose Damgaard, Rasmus Pedersen, and Thomas Bak

The paper has been submitted for journal publication and has been published as a preprint in
Preprints, no. 2022110511, 2022.
`https://doi.org/10.20944/preprints202211.0511.v1`

Abstract

Inspired by the reflective and deliberative control mechanisms used in cognitive architectures such as SOAR and Sigma, we propose an alternative decision mechanism driven by architectural appraisals allowing robots to overcome impasses. The presented work builds on and improves on our previous work on a generally applicable decision mechanism with roots in the Standard Model of the Mind and the Generalized Cognitive Hour-glass Model. The proposed decision mechanism provides automatic context-dependent switching between exploration-oriented, goal-oriented, and backtracking behavior, allowing a robot to overcome impasses. A simulation study of two applications utilizing the proposed decision mechanism is presented demonstrating the applicability of the proposed decision mechanism.

D.1 Introduction

Robotic technology has immense potential to change our daily life. In the industry, human-robot co-working is envisioned to play a key role in the next industrial revolution known as Industry 5.0 [1]. In healthcare robots also see increasing usage e.g. in personalized healthcare for providing assistance to patients, and the elderly [2, 3], and during the COVID-19 pandemic, robots were deployed to disinfect common spaces, such as supermarkets and hospitals [4]. Common to the above is the increased need for autonomous robots that can safely and naturally interact with humans while solving different abstractly and/or vaguely defined tasks. Due to the uncertainties in such problems, pure goal-driven problem-solving architectures will often end up in local minima in the problem formulation also known as impasses. I.e., situations where the information or action selection strategy currently available to the robot is insufficient to solve the task. Thus, one core faculty of such robotic systems should be the ability to reflect on the current situation to timely deviate from one action selection strategy to try out other strategies or to retrieve new information about the task.

The next generation of cognitive architectures, based on modern machine learning techniques, has the potential to revolutionize robotics by allowing roboticists to develop such autonomous systems easily. In previous work, we proposed the Generalized Cognitive Hour-glass Model constituting a framework for developing cognitive architectures by composing them from generally applicable probabilistic programming idioms over which powerful general algorithms can perform inference [5]. The idiomatic approach to composing cognitive architectures, encouraged by this framework, allows researchers and practitioners to more easily cooperate by mixing and matching probabilistic programming idioms developed by others while being able to handcraft parts of a system for which current solutions do not suffice.

In another work, we proposed one such probabilistic programming idiom based on the "standard model of the mind" [6] for the task of Active Knowledge Search (AKS) in unknown environments [7]. This idiom defines a probabilistic decision process that encourages a robot to take actions to discover, i.e. obtain information about, its environment based purely on notions of progress and information gain while avoiding constraint violations. Simulations applying this idiom to the specific problem of active mapping and robot exploration showed promising results. However, limitations were also identified. The main limitation was that in specific situations the simulated robot would get "stuck" taking repetitive actions yielding no new information about the environment, thus hindering full exploration of the environment. As we will discuss in more detail in Section D.4, this is essentially caused by the fixed strategy for action selection employed by the previous solution.

In the literature related to robot navigation, similar phenomena are commonly known as "the local minima issue" [8], "deadlocks" [9], "limit cycles" [10], "infinite loops" [11], "dead ends", "cyclic dead ends", or "trap-situations" [12]. Like the problem mentioned above, all of these terms refer to situations in which a fixed strategy for action selection results in no meaningful progress towards a goal state or compared to a measure of optimality. To resolve these situations solutions proposed by researchers within robotics usually rely on problem-specific information, e.g. geometric properties, to detect and/or resolve the impasse. As an example consider the approach used in [13] where a grid map is defined over the workspace with a counter attached to each of the cells keeping track of the number of times a given cell has been visited. Whenever this counter reaches a predefined threshold, it is registered as a limit cycle. When a limit cycle is detected a temporary way-point is generated, guiding the robot out of the enclosure causing the limit cycle. Finally, when the robot gets outside the enclosure, a virtual wall is generated, ensuring that the robot does not enter the problematic enclosure again. As another example consider the approach used in [14] where deadlock loops are detected based on the periodicity of the distance to the goal. Whenever a deadlock loop is detected, the distance to the goal is stored, and a wall-following behavior is initiated until an escape condition has been met. Similarly, in [15], deadlocks are detected based on a preferred velocity magnitude, the actual velocity magnitude, and the unsigned distance between robots. Whenever a deadlock is detected a deadlock resolution strategy is initiated. While the solutions suggested above might work for specific problems, they do not easily generalize to other problems.

In the literature related to cognitive architectures, similar phenomena in which an agent is unable to make progress with the information that is currently available are often referred to as impasses [16, 17]. As we will elaborate upon in Section D.2 research in cognitive architectures is focused on generally applicable solutions opposite to the problem-specific solutions commonly proposed in the robotics literature. Nevertheless, solutions seem to follow the same pattern as those proposed by researchers in robotics. First, systems are made able to detect impasses. Secondly, systems are induced with some sort of reflective mechanism that based on the detected impasse can choose appropriate temporary decision strategies until the impasse has been resolved. However, as we will also elaborate upon in Section D.2 the approaches taken by some of the most prominent cognitive architectures, SOAR [16] and Sigma [17], usually requires that controls are abstracted to symbolic representations that have to be decoded by an extra module external to the decision process. Thus, the fine level of control needed within robotics cannot be accounted for as an intrinsic part of the decision process.

For these reasons, our intention with this paper is to present our recent efforts toward implementing general reflective mechanisms similar to the ones found in cognitive architectures within the scope of the framework proposed in [5] in a way that is suitable for robotic applications. The main contributions of this paper are:

- A description and implementation of a control structure grounded in stochastic variational inference that is capable of deliberate and reflective control based on architectural appraisals.
- A demonstration of how such a control structure overcomes the limitations of the probabilistic programming idiom previously proposed in [7].
- A demonstration of how such general control structure compares to problem-specific approaches commonly used in robotics.
- A discussion of the time complexity of the proposed control structure.

This paper is organized as follows: Section D.2 reviews how some of the most prominent cognitive architectures have tackled impasse phenomena. Section D.3 introduces the notation used within this paper. Section D.4 shortly describes the previously proposed probabilistic programming idiom in more detail together with the impasse phenomenon observed. Modifications and extensions to the previously proposed idiom are presented in Section D.5 and Section D.6. Simulation results utilizing the modifications are provided in Section Section D.7. Section D.8 concludes the paper, and potential future directions are given in this section.

D.2 Impasses in SOAR and Sigma

Two of the most prominent cognitive architectures, SOAR [16] and Sigma [17], are both based on the problem space computational model [18], suggesting that problem spaces can be specified in terms of sets of states, S, and

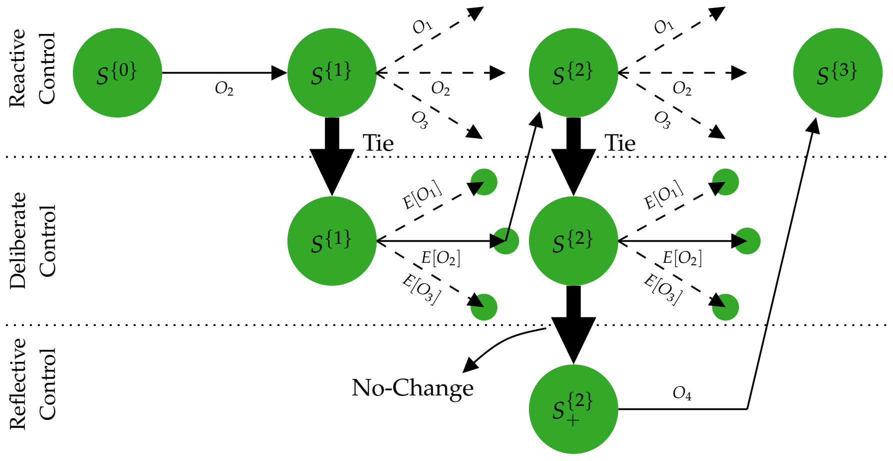


Fig. D.1: Illustration of the tri-level control structure employed by the two cognitive architectures SOAR [16] and Sigma [17]. For the state at time $t = 0$, S^0, only a single operator is relevant and a *reactive control* mechanism thus chooses to effectuate this operator which leads to a new state, S^1. In this new state, S^1, multiple operators, O_1, O_2, O_3, are proposed by the *reactive control* mechanism. Since no preference exists for these operators a tie impasse is detected, and the *deliberate control* mechanism simulates the expected outcome, $E(O_i)$, from effectuating each of the operators. In this state, S^1, the *deliberate control* mechanism is able to select an operator, O_2, based on preferences for the expected outcome, leading to a new state S^2. In S^2 a tie impasse also occurs, however, this time the expected result of applying any of the proposed operators, O_1, O_2, O_3, results in the next state, S^3, being similar to the current state, i.e. $S^3 = S^2$. Therefore, a no-change impasse is detected, and a *reflective control* mechanism is activated which changes the current state in a way such that a new operator, O_4, can be proposed.

operators, O, and that goals can be reached by knowledge search in such problem spaces. However, both architectures acknowledge that direct knowledge search might not always be possible e.g., due to insufficient or ambiguous knowledge. Therefore, these architectures implement a nested tri-level control structure with higher levels activated by the detection of impasses as illustrated in Fig. D.1 [17]. At the base level, a *reactive control* mechanism proposes operators relevant to the current state based on available knowledge. If only one relevant operator is proposed this operator is effectuated. If multiple relevant operators, $O_1, ..., O_i$ are proposed it is detected as a *tie impasse* and the *deliberative control* mechanism can evaluate the expected outcome resulting from applying each of the proposed operators to the current state. Based on these outcomes the *deliberative control* mechanism might be able to settle on one of the relevant operators to effectuate. By controlling how the expected outcome influences the decision making the *deliberative control* mechanism can form sequential, knowledge-driven, or algorithmic behavior. Finally, whenever the *deliberative control* mechanism cannot pick a unique operator other types of impasses are detected. The impasses detected depend on the reason why no operator could be picked. Based on these impasses a *reflective control* mechanism can take additional actions in order to solve the impasse. The actions taken by the *reflective control* mechanism depend on the specific impasse detected and can include the generation of alternative subgoals, the inclusion of additional information into the problem space, or even the generation of additional and entirely different problem spaces e.g., to perform meta-reasoning. Table D.1 summarizes the different types of impasses detected by the two cognitive architectures SOAR and Sigma.

Table D.1: Overview of the different types of impasses detected by the two cognitive architecture SOAR and Sigma.

Impasse	**SOAR** [16]	**Impasse**	**Sigma** [19]
Operator Tie	Occurs when multiple operators are proposed without any preference being able to select between them	**Tie**	Occurs when there are multiple candidate operators, but available knowledge is insufficient to choose among them.
Operator Conflict	Occurs when preferences for two proposed operators, O_1 and O_2, indicates that $O_1 \succ O_2$ and $O_2 \succ O_1$	**No-Change**[1]	Occurs when an operator is selected but no state change results.
Operator No-Change	An operator remains selected for consecutive decision cycles	**None**	Occurs when there are no candidate operators for selection.
State No-Change	No acceptable preferences or every candidate operator also has a reject preference		

[1] According to the description given in [19], however, in [17] it is stated that a no-change impasse occurs "when an operator remains selected for more than one cycle".

To summarize, in these tri-level control structures the *reflective control* mechanism uses the *deliberative control* mechanism as its inner loop which in turn uses the *reactive control* mechanism as its own inner loop. This approach assumes that there a priori exists a discrete/symbolic set of operators that re-actively can be either sorted out or proposed for further evaluation, thereby making detection of the impasses straightforward without any problem-specific knowledge. This approach has some clear benefits with respect to attention, i.e., the effective allocation of limited (computational) resources. Since each of the layers focuses computations on the information actually needed to solve a given problem in a specific context/state a lot of computations are saved. This is especially true when the approach is coupled with Appraisal Theory [20, 21], such that the evaluations of operators initiated by the *deliberative control* mechanism are grounded in a limited set of appraisals variables. However, the approach also raises some difficulties in robotics, where a lot of the low-level control is more naturally described by means of continuous variables. As an example, consider the position control of a robot. In SOAR and Sigma, such controls are usually abstracted to symbolic representations such as "walk towards target object", "run towards target object", "pick up target object" or "walk towards random object" [17]. These symbolic representations then have to be decoded by an extra module external to the decision process, called the *motor buffer*, before they can be manifested in the environment. This layer of abstraction makes it hard to incorporate uncertainties resulting from low-level control into the decision process, which in the end will result in less optimal responses being picked.

D.3 Preliminaries

As in paper [7] we use the following notation. X is used to denote observed variables, Z is used to denote latent variables, and C is used to denote a collection of both types of variables. A superscript in curly brackets is used to indicate the index of a variable. For time indexes, the set of indexes of future variables is indicated as $\{t\}^{+} = \{t+1, ..., t+\overline{T}\}$. Similarly, the set of indexes of past variables is indicated as $\{t\}^{-} = \{t-\underline{T}, ..., t\}$. Furthermore, within this paper the following approximate "probabilistic logic"

$$\begin{aligned}
&p(z \in \overline{z} \vee y \in \overline{y}) \overset{\text{def}}{=} p(z \in \overline{z}) + p(y \in \overline{y}) - p(z \in \overline{z})p(y \in \overline{y}) \\
&p(z \in \overline{z} \wedge y \in \overline{y}) \overset{\text{def}}{=} p(z \in \overline{z}) \cdot p(y \in \overline{y}) \\
&p\left(\bigwedge_{i=1}^{I} z^{\{i\}} \in \overline{z}^{\{i\}}\right) \overset{\text{def}}{=} \prod_{i=1}^{I} p\left(z^{\{i\}} \in \overline{z}^{\{i\}}\right)
\end{aligned}$$

is used, where $\wedge$ and $\vee$ denotes an approximate *and* and *or* operation, respectively. These approximate "probabilistic logic" rules constitute a probabilistic intersection and union with an independence assumption implied, respectively.

D.4 Problem elaboration

As stated in Section D.1, the probabilistic programming idiom proposed in [7] defines a probabilistic decision process for Active Knowledge Search in unknown environments, based on the "standard model of the mind" [6]. This was done by first defining a probabilistic model relating the previous content of working memory, $Z_{\text{WM}}^{\{t\}^-}$, with the future content, $C_{\text{WM}}^{\{t\}^+}$, while taking variables stored in the long-term memory, Z_{LTM}, into account. In [7], the working memory was further sub-divided into variables relating to motoric actions i.e., the *motor buffer*, Z_{Mb}, variables related to the *perceptual buffer*, Z_{Pb}, *State variables*, Z_{s}, representing the state of the agent itself, the environment, and *decision variables* $C_{\text{D}}^{\{t\}^+}$. From this a probabilistic decision model with the factorization

$$\begin{aligned}
&p\left(C_{\text{WM}\backslash\text{b}}^{\{t\}^+}, Z_{\text{Mb}}^{\{t-1\}^+}, Z_{\text{Pb}}^{\{t\}^+} | Z_{\text{WM}\backslash\text{b}}^{\{t\}^-}, Z_{\text{LTM}}\right) \qquad \text{(D.1)}\\
&\overset{def}{=} \left[\prod_{\tau=t+2}^{t+\overline{T}} p\left(C_{\text{D}}^{\{\tau\}} | Z_{\text{s}}^{\{\tau\}}, Z_{\text{WM}\backslash\text{b}}^{\{t\}^-}, Z_{\text{LTM}}\right) p\left(Z_{\text{s}}^{\{\tau\}} | Z_{\text{s}}^{\{\tau-1\}}, Z_{\text{Mb}}^{\{\tau-1\}}\right) p\left(Z_{\text{Mb}}^{\{\tau-1\}}\right)\right]\\
&\quad \cdot p\left(C_{\text{D}}^{\{t+1\}} | Z_{\text{s}}^{\{t+1\}}, Z_{\text{WM}\backslash\text{b}}^{\{t\}^-}, Z_{\text{LTM}}\right) p\left(Z_{\text{s}}^{\{t+1\}} | Z_{\text{s}}^{\{t\}}, Z_{\text{Mb}}^{\{t\}}\right) p\left(Z_{\text{Mb}}^{\{t\}}\right)
\end{aligned}$$

were derived, where $Z_{\text{WM}\backslash\text{b}} = Z_{\text{WM}} \backslash \{Z_{\text{Mb}}, Z_{\text{Pb}}\}$. Inspired by the work on emotions in [22], a subset of the decision variables, x_{A}, was denoted *attention variables*. The purpose of these *attention variables* is to control how the decision process is influenced by the other decision variables: *progress*, z_p, *information gain*, z_i, and *constraints*, z_c, hereafter referred to as *appraisal variables*. In [7] this was done via the fixed relation

$$\begin{aligned}
&p\left(x_A^{\{\tau\}} | Z_{\text{s}}^{\{\tau\}}, Z_{\text{WM}\backslash\text{b}}^{\{t\}^-}, Z_{\text{LTM}}\right) \qquad \text{(D.2)}\\
&= \mathit{Bernoulli}\left(p\left(\left[z_{\text{p}}^{\{\tau\}} = 1 \vee z_{\text{i}}^{\{\tau\}} = 1\right] \wedge z_{\text{c}}^{\{\tau\}} = 1 \middle| Z_{\text{s}}^{\{\tau\}}, Z_{\text{WM}\backslash\text{b}}^{\{t\}^-}, Z_{\text{LTM}}\right)\right)
\end{aligned}$$

which basically states that during the decision process attention should be given to future states that yield progress *or* new information *and* does not violate constraints. Having defined the model in Eq. (D.1) and the relation in Eq. (D.2), Stochastic Variational Inference was used to approximate the posterior over optimal future motoric actions given the attention variables, i.e.,

$$q_{\phi_{\text{Mb}}^{\{t-1\}^+,*}}\left(Z_{\text{Mb}}^{\{t-1\}^+}\right) \approx p\left(Z_{\text{Mb}}^{\{t-1\}^+} | x_{\text{A}}^{\{t\}^+} = 1\right) \qquad \text{(D.3)}$$

The above was implemented as an abstract class utilizing the probabilistic programming language Pyro [23], thereby ensuring that the probabilistic programming idiom can be reused in multiple applications by implementing a few abstract methods defined by the abstract class.

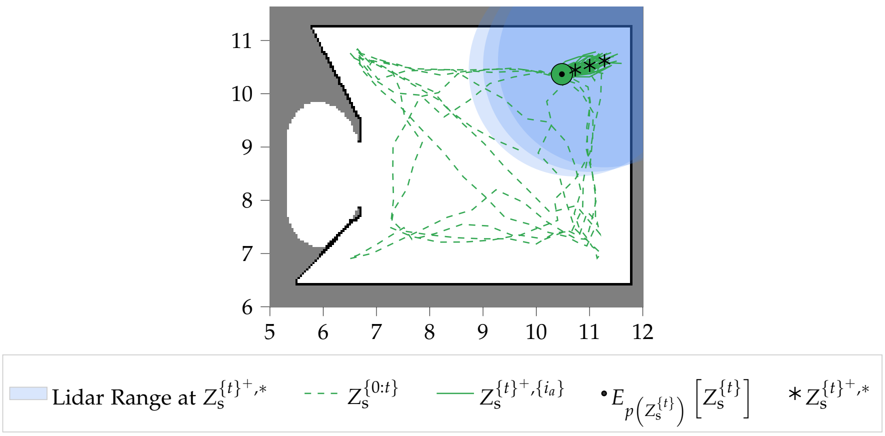


Fig. D.2: A simulated trajectory of using the method presented in [7] for the floor plan with ID "0a1b29dba355df2ab02630133187bfab" from the HouseExpo dataset [24]. The robot keeps driving around in the same room, without exploring the rest of its environment.

To investigate the performance of the idiom, it was used to implement an algorithm for autonomous robot exploration which was simulated on the full HouseExpo dataset [24], containing 35126 different floor plans. From these simulations, one of the observations was that the robot sometimes would end up taking repetitive actions purely driven by the *progress* appraisal variable. Whereby, the robot would not fully explore its environment as illustrated in Fig. D.2. In other words, the robot ended up at an impasse. It was further concluded that an alternative to the fixed decision strategy given by Eq. (D.2) would be needed to overcome this problem.

D.5 Overall Idea

Even though both SOAR and Sigma are said to implement a tri-level control structure, the distinction between the *deliberative control* and the *reflective control* mechanisms seems architectural rather than conceptual. By that, we mean that they both simply comprise a specific architectural response to similar architectural stimuli, i.e., the detection of impasses. When we further consider the statement:

"Work in Sigma on appraisal, and its relationship to attention, has led to the conclusion that the detection of impasses should itself be considered as a form of appraisal" [17].

It hints toward the possibility that similar functionality might be obtained from an architecturally simpler control structure. Based on this, and to over-

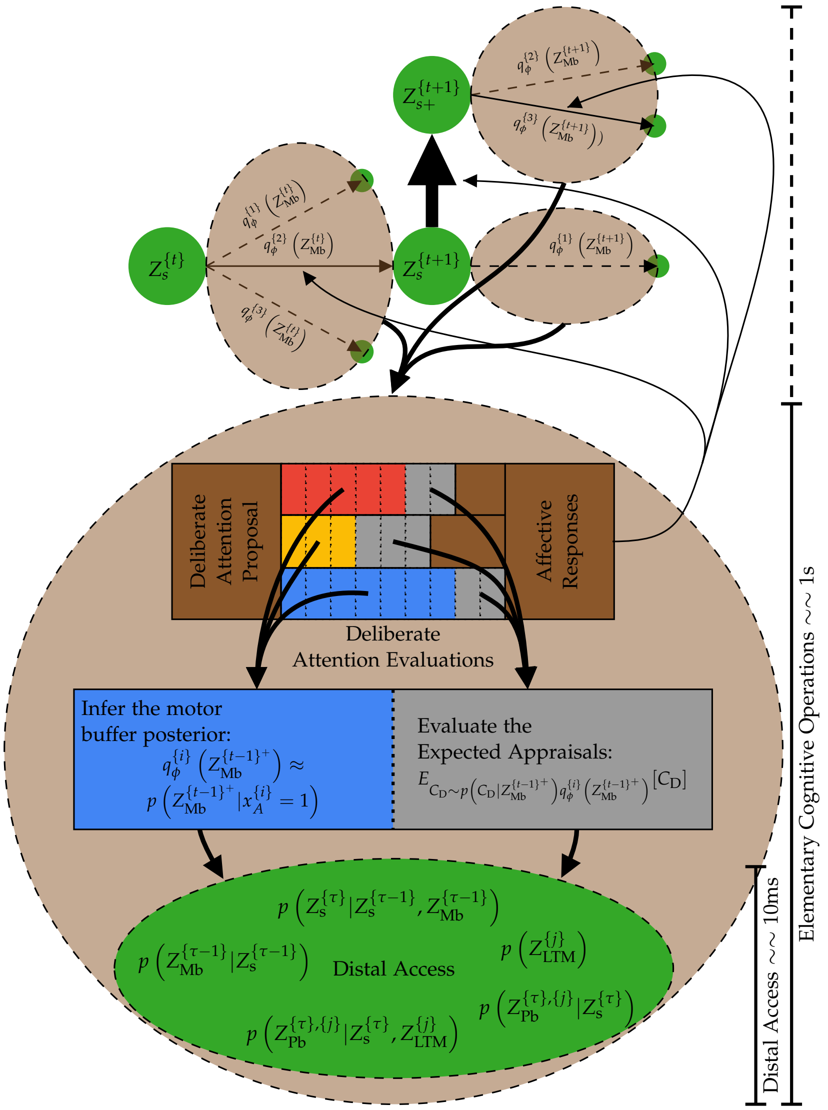

$q_\phi^{\{2\}}\left(Z_{\mathrm{Mb}}^{\{t+1\}}\right)$
$Z_{s+}^{\{t+1\}}$
$q_\phi^{\{3\}}\left(Z_{\mathrm{Mb}}^{\{t+1\}}\right))$
$q_\phi^{\{1\}}\left(Z_{\mathrm{Mb}}^{\{t\}}\right)$
$q_\phi^{\{2\}}\left(Z_{\mathrm{Mb}}^{\{t\}}\right)$
$Z_s^{\{t\}}$
$Z_s^{\{t+1\}}$
$q_\phi^{\{1\}}\left(Z_{\mathrm{Mb}}^{\{t+1\}}\right)$
$q_\phi^{\{3\}}\left(Z_{\mathrm{Mb}}^{\{t\}}\right)$
Deliberate Attention Proposal
Affective Responses
Deliberate Attention Evaluations
Infer the motor buffer posterior: $q_\phi^{\{i\}}\left(Z_{\mathrm{Mb}}^{\{t-1\}^+}\right) \approx p\left(Z_{\mathrm{Mb}}^{\{t-1\}^+} | x_A^{\{i\}} = 1\right)$
Evaluate the Expected Appraisals: $E_{C_D \sim p\left(C_D | Z_{\mathrm{Mb}}^{\{t-1\}^+}\right) q_\phi^{\{i\}}\left(Z_{\mathrm{Mb}}^{\{t-1\}^+}\right)}[C_D]$
$p\left(Z_s^{\{\tau\}} | Z_s^{\{\tau-1\}}, Z_{\mathrm{Mb}}^{\{\tau-1\}}\right)$
$p\left(Z_{\mathrm{Mb}}^{\{\tau-1\}} | Z_s^{\{\tau-1\}}\right)$
Distal Access
$p\left(Z_{\mathrm{LTM}}^{\{j\}}\right)$
$p\left(Z_{\mathrm{Pb}}^{\{\tau\},\{j\}} | Z_s^{\{\tau\}}\right)$
$p\left(Z_{\mathrm{Pb}}^{\{\tau\},\{j\}} | Z_s^{\{\tau\}}, Z_{\mathrm{LTM}}^{\{j\}}\right)$
Distal Access $\sim\sim$ 10ms
Elementary Cognitive Operations $\sim\sim$ 1s

Fig. D.3: Illustration of the proposed approach with an indication of where we consider each element of the approach to fit into the approximate timescales at which humans make decisions set forward by Allen Newell in [25].

come the limitations of the approach described in Section D.2, we propose an alternative approach centered around appraisals. As illustrated in Fig. D.3, our proposal is to have a control structure consisting of a single architectural layer with decisions being the result of three main steps:

- Deliberate Attention Proposal
- Deliberate Attention Evaluations
- Affective Responses

Considering the tri-level control structure described in Section D.2 this resembles a combination of the deliberate and reflective mechanism. The main difference is that here attention mechanisms for choosing motoric actions similar to Eq. (D.2) are proposed for evaluation rather than operators for which the outcome is known a priori. Each of the proposed deliberate attention mechanisms might consider a subset of and/or special combinations and weightings of the appraisal variables available to the robot. Thereby, promoting different behaviors. The posterior distributions describing the specific motoric actions corresponding to these proposed deliberate attention mechanisms first become available to the decision process as part of the *Deliberate Attention Evaluations* step. To obtain the posterior over motoric actions the *Deliberate Attention Evaluations* step follows the same steps described in Section D.4 for each of the attention mechanisms proposed by the *Deliberate Attention Proposal* step. Besides inferring the motoric action posterior the *Deliberate Attention Evaluations* step also evaluates what the expected appraisals would be from effectuating each of them. Based on the expected appraisals of each of the motoric action posteriors the last step in the decision process can initiate different *affective responses*, such as effectuating one of the action posteriors or proposing additional attention mechanisms to evaluate. Thus, instead of treating the detection of and responses to impasses as distinctive architectural mechanisms, we propose that this is treated as *affective responses* to appraisals evaluated during the *Deliberate Attention Evaluations* step. While the proposed approach conceptually does support deliberate and reflective responses via the *affective responses*, it does not currently have support for reactive responses, since all motoric actions have to be inferred from the deliberate attention mechanisms. However, in Section D.8 we will discuss how we imagine that reactive responses could be incorporated into the control structure. Furthermore, modern probabilistic programs such as Pyro [23] can combine stochastic variational inference with enumeration to infer the motoric action posterior. Thereby, making it possible to combine operators represented by both discrete/symbolic and continuous variables in the proposed control structure.

D.6 Idiom Modifications and Extensions

To test the approach proposed in Section D.5 several modifications and extensions were made to the probabilistic programming idiom proposed in [7].

This includes additional appraisal variables, the possibility of adding and using additional deliberate attention mechanisms, together with an implementation of simple mechanisms for *deliberate attention proposal* and *affective responses*. In order to make the implementation reusable in the spirit of the framework presented in [5], all of this is implemented as a series of abstract python classes each constituting a probabilistic programming idiom available at [26].

D.6.1 Additional Appraisal Variables

Besides the *progress*, z_p, *information gain*, z_i, and *constraints*, z_c, appraisal variables defined in [7], a couple of new appraisal variables has been implemented. The first was the *accummulated constraints* appraisal,

$$\begin{aligned} p\left(z_{\text{Ac}} = 1 \,\middle|\, Z_{\text{s}}^{\{t+1:\overline{T}\}}, Z_{\text{WM}\backslash\text{b}}^{\{t\}^-}, Z_{\text{LTM}}\right) \quad (D.4) \\ = Bernoulli\left(p\left(\bigwedge_{\tau=t+1}^{\overline{T}} z_{\text{c}}^{\{\tau\}} = 1 \,\middle|\, Z_{\text{s}}^{\{t+1:\overline{T}\}}, Z_{\text{WM}\backslash\text{b}}^{\{t\}^-}, Z_{\text{LTM}}\right)\right), \end{aligned}$$

which was implemented due to a need to check constraint violations of a full state trajectory rather than at a single state. The second originated from a need to be able to define desirable/goal states that a robot should seek to attain. First, we approximate the KL-divergence between a desirable state, Z_{s}^*, and the state, $Z_{\text{s}}^{\{\tau\}}$, after effectuating the motoric action, $Z_{\text{Mb}}^{\{\tau-1\}}$, as

$$\begin{aligned} & D_{\text{KL}}\left[p\left(Z_{\text{s}}^*\right) || p\left(Z_{\text{s}}^{\{\tau\}} | Z_{\text{s}}^{\{\tau-1\}}, Z_{\text{Mb}}^{\{\tau-1\}}\right)\right] \\ & = E_{\hat{Z}_{\text{s}}^{\{\tau\}}}\left[log\left(\frac{p\left(Z_{\text{s}}^{\{\tau\}} = \hat{Z}_{\text{s}}^{\{\tau\}} | Z_{\text{s}}^{\{\tau-1\}}, Z_{\text{Mb}}^{\{\tau-1\}}\right)}{p\left(Z_{\text{s}}^* = \hat{Z}_{\text{s}}^{\{\tau\}}\right)}\right)\right] \\ & \approx \frac{1}{I}\sum_{i=1}^{I}\left(\begin{array}{l} log\left(p\left(Z_{\text{s}}^* = \hat{Z}_{\text{s}}^{\{\tau\},\{i\}}\right)\right) \\ \quad -log\left(p\left(Z_{\text{s}}^{\{\tau\}} = \hat{Z}_{\text{s}}^{\{\tau\},\{i\}} | Z_{\text{s}}^{\{\tau-1\}}, Z_{\text{Mb}}^{\{\tau-1\}}\right)\right) \end{array}\right) \quad (D.5) \\ & \approx ReLu\left(log\left(p\left(Z_{\text{s}}^* = \hat{Z}_{\text{s}}^{\{\tau\}}\right)\right) - log\left(p\left(Z_{\text{s}}^{\{\tau\}} = \hat{Z}_{\text{s}}^{\{\tau\}} | Z_{\text{s}}^{\{\tau-1\}}, Z_{\text{Mb}}^{\{\tau-1\}}\right)\right)\right) \\ & \quad \stackrel{\text{def}}{=} D_{Z_{\text{s}}^*}\left(\hat{Z}_{\text{s}}^{\{\tau\}}\right). \end{aligned}$$

Inspired by the optimality variable defined in [27] we then define the *desirability* appraisal, $z_{\text{d},Z_{\text{s}}^*}^{\{\tau\}}$, as

$$p\left(z_{\text{d},Z_{\text{s}}^*}^{\{\tau\}} = 1 \,\middle|\, Z_{\text{s}}^{\{t+1:\overline{T}\}} = \hat{Z}_{\text{s}}^{\{t+1:\overline{T}\}}, Z_{\text{WM}\backslash\text{b}}^{\{t\}^-}, Z_{\text{LTM}}\right) \quad (D.6)$$

$$= \begin{cases} 0,0 & ;\text{if } p\left(z_{\text{Ac}} = 1 \middle| \begin{matrix} Z_{\text{s}}^{\{t+1:\overline{T}\}}, \\ Z_{\text{WM}\setminus\text{b}}^{\{t\}^-}, Z_{\text{LTM}} \end{matrix}\right) < 1 \\ \textit{Bernoulli}\left(-e^{-\sigma_d \cdot D_{Z_{\text{s}}^*}\left(\hat{Z}_{\text{s}}^{\{\tau\}}\right)}\right) & ;\ \textit{else} \end{cases}$$

where the subscript Z_{s}^* in $z_{\text{d},Z_{\text{s}}^*}^{\{\tau\}}$ is used to denote the dependency on $p\left(Z_{\text{s}}^*\right)$, and σ_d is a scaling factor. Equation (D.6) defines a pseudo probability for which states most similar to the desirable state, Z_{s}^*, has the highest probability, and states that are less similar have an exponentially lower probability, while states resulting from trajectories that violate constraints have zero probability. The dependency on the *accummulated constraint* appraisal was introduced to aid in overcoming a small probability of constraint violation observed in [7]. For the same reason the *progress*, z_p, and *information gain*, z_i, appraisals has also been modified as follows

$$p\left(z_i^{\{\tau\}} = 1 \middle| Z_{\text{s}}^{\{t+1:\overline{T}\}}, Z_{\text{WM}\setminus\text{b}}^{\{t\}^-}, Z_{\text{LTM}}\right)$$
$$= \begin{cases} 0,0 & ;\text{if } p\left(z_{\text{Ac}} = 1 \middle| \begin{matrix} Z_{\text{s}}^{\{t+1:\overline{T}\}}, \\ Z_{\text{WM}\setminus\text{b}}^{\{t\}^-}, Z_{\text{LTM}} \end{matrix}\right) < 1 \\ p\left(z_{\tilde{i}}^{\{\tau\}} = 1 \middle| \begin{matrix} Z_{\text{s}}^{\{t+1:\overline{T}\}}, \\ Z_{\text{WM}\setminus\text{b}}^{\{t\}^-}, Z_{\text{LTM}} \end{matrix}\right) & ;\ \textit{else} \end{cases}$$

$$p\left(z_p^{\{\tau\}} = 1 \middle| Z_{\text{s}}^{\{t+1:\overline{T}\}} = \hat{Z}_{\text{s}}^{\{t+1:\overline{T}\}}, Z_{\text{WM}\setminus\text{b}}^{\{t\}^-}, Z_{\text{LTM}}\right)$$
$$= \begin{cases} 0,0 & ;\text{if } p\left(z_{\text{Ac}} = 1 \middle| \begin{matrix} Z_{\text{s}}^{\{t+1:\overline{T}\}}, \\ Z_{\text{WM}\setminus\text{b}}^{\{t\}^-}, Z_{\text{LTM}} \end{matrix}\right) < 1 \\ p\left(z_{\tilde{p}}^{\{\tau\}} = 1 \middle| \begin{matrix} Z_{\text{s}}^{\{t+1:\overline{T}\}}, \\ Z_{\text{WM}\setminus\text{b}}^{\{t\}^-}, Z_{\text{LTM}} \end{matrix}\right) & ;\ \textit{else} \end{cases}$$

where $z_{\tilde{p}}$ and $z_{\tilde{i}}$ are the *progress*, z_p, and *information gain*, z_i, appraisals as defined in [7].

D.6.2 Deliberate Attention mechanisms

Based on the appraisals defined in Section D.6.1 five different deliberate attention mechanisms have been implemented. All these can be defined as

$$p\left(x_A^{\{\tau\}} | Z_{\text{s}}^{\{t+1:\overline{T}\}}, Z_{\text{WM}\setminus\text{b}}^{\{t\}^-}, Z_{\text{LTM}}\right)$$
$$= \textit{Bernoulli}\left(p\left(\Phi(\tau) \middle| Z_{\text{s}}^{\{t+1:\overline{T}\}}, Z_{\text{WM}\setminus\text{b}}^{\{t\}^-}, Z_{\text{LTM}}\right)\right)$$

where $\Phi(\tau)$ defines the logic for combining appraisals as in Table D.2. Each of these deliberate attention mechanisms promotes different behaviors.

Table D.2: Definitions of $\Phi(\tau)$ used for each of the implemented deliberate attention mechanisms.

Deliberate Attention Mechanism	$\Phi(\tau)$ for $\tau \in [t+1; \overline{T}-1]$	$\Phi(\overline{T})$ for $\tau = \overline{T}$
ConstraintAvoidance - CA	$z_{\text{Ac}} = 1$	$z_{\text{Ac}} = 1$
StateReach - SR(Z_{s}^*)	$z_{\text{Ac}} = 1$	$\begin{cases} z_{\text{Ac}} = 1 & ; P(z_{\text{Ac}} = 1) < 1 \\ z_{\text{d},Z_{\text{s}}^*}^{\{\overline{T}\}} = 1 & ;\ else \end{cases}$
StateReachWithProgress - SRP(Z_{s}^*)	$z_{\text{Ac}} = 1$	$\begin{cases} z_{\text{Ac}} = 1 & ; P(z_{\text{Ac}} = 1) < 1 \\ z_{\text{d},Z_{\text{s}}^*}^{\{\overline{T}\}} = 1 \wedge z_{p}^{\{\overline{T}\}} = 1 & ;\ else \end{cases}$
StateReachWithExplore - SRE(Z_{s}^*)	$z_{\text{Ac}} = 1$	$\begin{cases} z_{\text{Ac}} = 1 & ; P(z_{\text{Ac}} = 1) < 1 \\ z_{\text{d},Z_{\text{s}}^*}^{\{\overline{T}\}} = 1 \wedge z_{i}^{\{\overline{T}\}} = 1 & ;\ else \end{cases}$
Explore - E	$z_{\text{Ac}} = 1$	$\begin{cases} z_{\text{Ac}} = 1 & ; P(z_{\text{Ac}} = 1) < 1 \\ z_{i}^{\{\overline{T}\}} = 1 & ;\ else \end{cases}$
ExploreWithProgress - EP	$z_{\text{Ac}} = 1$	$\begin{cases} z_{\text{Ac}} = 1 & ; P(z_{\text{Ac}} = 1) < 1 \\ z_{i}^{\{\overline{T}\}} = 1 \wedge z_{p}^{\{\overline{T}\}} = 1 & ;\ else \end{cases}$

D.6.3 Deliberate Attention Proposal And Affective Responses

Based on the *deliberate attention mechanisms* defined in Section D.6.2, an *affective response* mechanism has been implemented with the purpose of making a robot effectively explore its environment and possibly navigate towards a goal state, Z_s^*, if defined. This *affective response* mechanism can be sub-divided into three parts responsible for different types of behaviors with pseudo-code given in Algorithm D.1, Algorithm D.2, and Algorithm D.3. Algorithm D.1 yields behavior that strives for the goal state. Algorithm D.2 yields behavior that strives to obtain new information about the environment. Finally, Algorithm D.3 yields behavior that strives to backtrack. The combined affective response only depends on the appraisals defined in Section D.6.1 and [7] which requires no problem-specific information, thereby, making this affec-

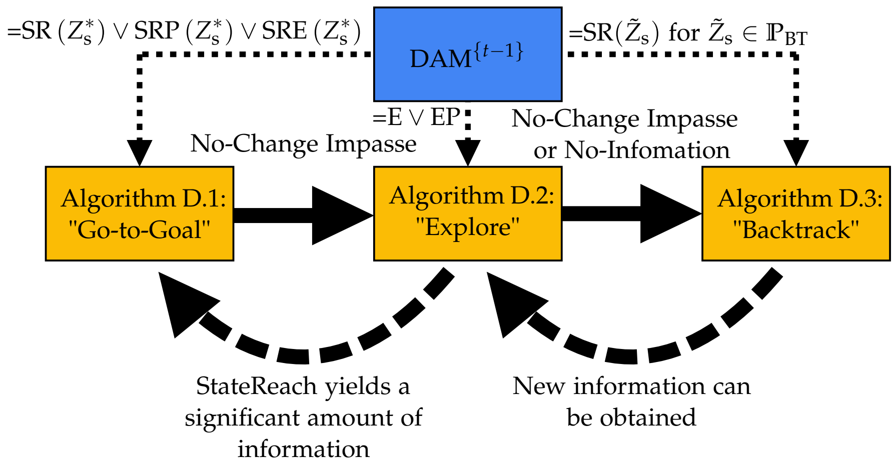

$=\text{SR}(Z_s^*) \vee \text{SRP}(Z_s^*) \vee \text{SRE}(Z_s^*)$
$\text{DAM}^{\{t-1\}}$
$=\text{SR}(\tilde{Z}_s)$ for $\tilde{Z}_s \in \mathbb{P}_{\text{BT}}$
$=\text{E} \vee \text{EP}$
No-Change Impasse
No-Change Impasse or No-Infomation
Algorithm D.1: "Go-to-Goal"
Algorithm D.2: "Explore"
Algorithm D.3: "Backtrack"
StateReach yields a significant amount of information
New information can be obtained

Fig. D.4: Overview of the implemented affective response mechanism sub-divided into 3 algorithms. Dotted lines indicate what algorithm is activated based on the which *deliberate attention mechanism*, $\text{DAM}^{\{t-1\}}$, resulted in the effectuation of a motoric action in the last time step. Solid lines indicate a reflective response resulting in a direct transition between the algorithms. Each direct transition requires a new *deliberate attention proposal* and *deliberate attention evaluation*. A dashed line indicates an indirect transition between the algorithms which takes effect in the next decision cycle. Z_s^* denotes a goal state, and $\mathbb{P}_{\text{BT}}$ denotes a path of previous states to backtrack.

tive response mechanism general and reusable. As illustrated in Fig. D.4, each part of the *affective response* mechanism is activated either as a reflective response to another part of the *affective response* mechanism or based on the *deliberate attention mechanism* that caused a motoric action to be effectuated in the last decision cycle. E.g. if the *deliberate attention mechanism* "Explore (E)" caused the effectuation of $q_\phi^{\{E\}}\left(Z_{\text{Mb}}^{\{t-1\}}\right)$ at time $t-1$, then Algorithm D.2 will be activated first at time t. In cases where the motoric action was caused by the *deliberate attention mechanism* "ConstraintAvoidance (CA)", the same algorithm is simply activated again. The intuition behind this affective response mechanism is as follows. If a goal state is known and if it is possible to attain it with the current knowledge directly, this should have first priority. If this is not possible new information should be sought after until the goal state can be attained. Finally, if in a state where new information cannot be obtained, the system should be able to bring itself back to a previous state in which new information can be obtained via backtracking. To support this *affective response* mechanism a *deliberate attention proposal* mechanism has been implemented that simply proposes the *deliberate attention mechanisms* required for each part of the *affective response* mechanism. When combined this exemplifies how both deliberate and reflective responses can be implemented grounded in the appraisals defined in Section D.6.2. In particular, notice that something similar to the "no-change" impasse in SOAR and Sigma is obtained on the basis of the "Progress" appraisal in both Algorithm D.1 and Algorithm D.2.

D.7 Results

To test the effectiveness of the proposed approach two different simulation studies were performed. Both of these were done utilizing the Pseudo-SLAM simulator [24], and using the same implementation of abstract methods for the probabilistic programming idiom that was used in [7]. The exact parameters used for each of these simulations can be found at [26], which also contains scripts to replicate each of the experiments.

D.7.1 Pure Exploration

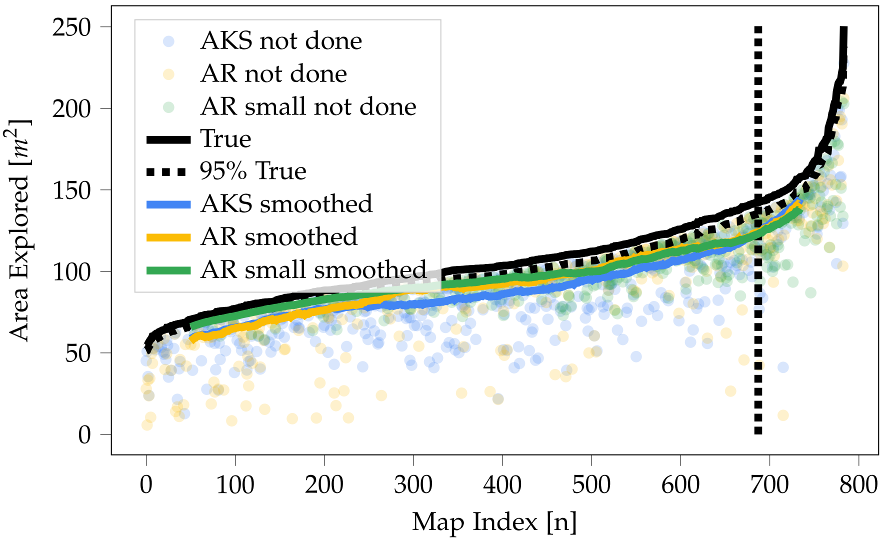


Fig. D.5: The area that the robot explored in each of the simulations. The indices of the 784 floorplans have been sorted by the true area of the map in ascending order. "AKS" shows results from using the method presented in [7]. "AR" shows results from using the *affective response* mechanism from Section D.6.3. The "smoothed" curves show a moving average with a window size of 100 and shifted 50 indexes. The "not done" scatter shows the exact area explored by the simulations in which the robot did not manage to explore 95% of the map or more.

The first simulation study was done in order to compare with results from [7]. In [7] we tested the exploration capability of the proposed algorithm by simulating it in 35126 floor plans from the HouseExpo dataset [24]. However, (1) many of these were so small that they were fully discovered in a few iterations, and (2) on the other end of the spectrum some of the floor plans were simply too big to be fully discovered within the maximum of 200 time-steps that was allowed in each simulation. Furthermore, (3) the problems of the previous solution discussed in Section D.4 are only noticeable in floor plans with more than one room. Additionally, (4) it was found that for some of the floor plans openings between the rooms were physically too small for the robot to squeeze through. Thus, for the purpose of efficiently testing the approach proposed in this paper we selected a smaller subset of the HouseExpo dataset satisfying the following criteria.

1. The floor plans should have a bounding box larger than 100 m^2 to avoid spending time on simulations redundant due to (1).
2. The floor plans should be fully discovered in the experiment from [7], in order to minimize the influence from (2) and (4).
3. The floor plans should contain more than 3 rooms in order to provoke (3).

Based on this, a subset of the HouseExpo dataset consisting of 784 floor plans where selected. Fig. D.5 and Table D.3 show the results of simulating our old approach, "AKS", again as well as the approach proposed within this paper, "AR". The simulations were performed with the same environmental and robot settings used in [7]. As no goal state was specified, only Algorithm D.2 and Algorithm D.3 were effectively used to drive the behavior of the "AR" method in these simulations. For each floor plan a random initial position where selected, and this initial position were utilized for both simulations. Notice, that even though one of the criteria for the selection of the subset of floor plans was that it should be fully explored by "AKS" in the experiment in [7], Fig. D.5 indicates that not all floor plans where fully explored by "AKS" in this new round of simulations. This is simply due to a difference in initial positions between simulations and illustrates a lack of robustness of "AKS". From Fig. D.5 it might seem that "AR" does not perform better than "AKS" for small floor plans. By visual inspection of the simulation trajectories, it was found that the reason for "AR" not being able to explore some floor plans fully was due to a lesser willingness to violate constraints compared to "AKS". This can be verified from the row "Collision pr. Timesteps" in Table D.3. This is especially pronounced in small floor plans, where the openings between rooms tend to be smaller. To further verify that the lack of exploration by "AR" is indeed due to its unwillingness to violate constraints, the third series of simulations denoted "AR small" was performed. In these simulations, the size of the robot, the uncertainty in its initial position, and the assumed motion uncertainty was decreased. By changing these parameters, it becomes easier for the robot to take actions through narrow openings without constraint violations. From Fig. D.5 it is seen that "AR small" fully explores nearly all of the small floor plans, and performs similarly to "AR"

Table D.3: Comparison of our approach with results for 6 different methods presented in [8].

Metric	**AKS**	**RGS**	**RGS small**
Maps Not Fully Explored	484	331	314
Mean Exploration Percentage	84,9%	87,3%	90,7%
Mean percentage explored for unfinished maps	78,6%	76,7%	84,4%
Maps with Collisions	18	0	0
Collisions	19	0	0
Collision pr. Timesteps	0,14‰	0,00‰	0,00‰

for all other maps as expected. As the floor plans get bigger, the ability of all three methods to fully explore the floor plans to a greater extent depends on initial conditions, rather than the ability of the methods to escape local minima. As a result, from Fig. D.5 it is observed that as the floor plans get larger all methods perform very similarly. Nevertheless, from Table D.3 it is evident that "AR" is indeed better for overcoming local minima and making the robot efficiently explore its environment.

D.7.2 Goal Seeking

The second series of simulations were performed in order to compare the proposed approach with more problem-specific approaches from [8]. To do so three of the test environments from [8] were recreated in the Pseudo-SLAM simulator as illustrated in Fig. D.6. These three environments are designed specifically with the purpose of causing local minima, and as such are perfect for testing the proposed approach. Since the approach proposed in this paper is based on probabilistic methods, some degree of variations in results should be expected. Therefore, 100 simulations were performed for each of these environments with the same initial conditions and goal state as in [8]. Since a goal state where specified for these simulations, the full capabilities of the *affective response* mechanism described in Section D.6.3 were effectively in use. Table D.4 summarizes the results from these simulations and compares them to the results from [8]. In all of the simulations, the robot managed to reach the goal state, thereby substantiating the ability of the proposed approach to escaping local minima. From Table D.4 it is furthermore seen that the "AR" method is better than any of the problem-specific methods in all three environments when only considering the minimum traveled distance. However,

Table D.4: The traveled distance in 3 different environments utilizing our approach, AR, compared with results for 6 other methods presented in [8]. As AR is based on probabilistic methods we ran 100 simulations and present the mean of the results together with the minimum and maximum values for each of the environments. The best result for each environment is highlighted with bold text.

Environment	**C-shaped**	**Double U-shaped**	**V-shaped**	**Average**
Random[1]	55	97	38	63,33
Reflected Virtual Target[1]	**45**	88	**27**	53,33
Global Path Backtracking[1]	59	100	29	62,67
Half Path Backtracking[1]	101	110	31	80,67
Local Path Backtracking[1]	59	96	28	61
Wall-Following[1]	1915	466	111	830,67
AR	45,34 [**39,38**-73,97]	**47,51** [**28,07**-72,56]	52,91 [**22,91**-114,98]	**48,59**

[1] Results from Table 2 in [8].

when considering the average distance for the "V-shape" environment it is nearly twice that of the best method from [8], i.e., "Reflected Virtual Target". Considering the first column of Fig. D.6 it is clear that the robot generally can take the two paths indicated with green and yellow colors. We suspect that the better average performance of the "Reflected Virtual Target" method in the "V-shape" environment is caused by an initial condition that makes the problem-specific methods favor paths similar to the one marked with yellow in Fig. D.6. The better performance achieved by such preference would not necessarily lead to better performance in general environments/problems, and a more reasonable comparison would probably be obtained by some variations in the initial conditions and/or goal state. As such we do not consider this an inauspicious characteristic of the "AR" approach.

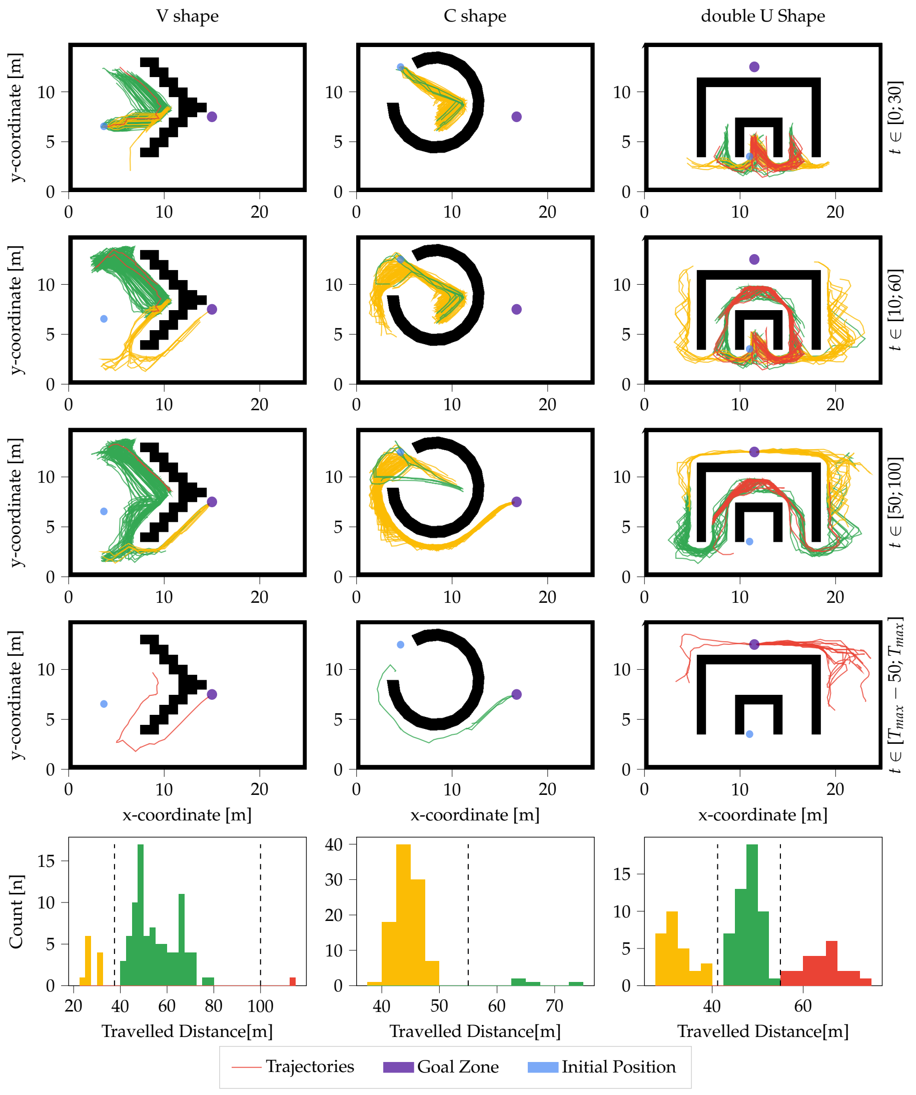


Fig. D.6: The robot trajectories in each of the 100 simulations for each of the three environments: "V shape", "C shape", and "double U shape". Each of the trajectories is color-coded according to the length of the trajectory. $T_{\max} = 308$, $T_{\max} = 191$, and $T_{\max} = 208$ for "V shape", "C shape", and "double U shape", respectively.

D.7.3 Timings

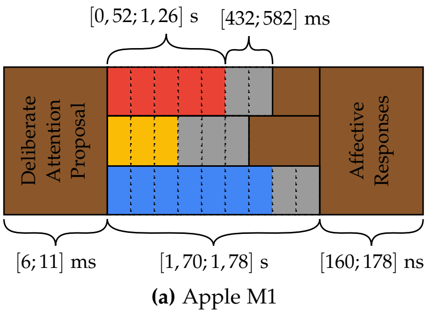


Fig. D.7: Timings for the main steps of the proposed solution running on two different CPUs, for the deliberate attention mechanism and the affective response mechanism presented in Section D.6.2 and Section D.6.3, respectively, and for the abstract method implementations specifically used for the simulations in Section D.7.1 and Section D.7.2.

One of the most critical features of any robotics system is the satisfaction of the real-time constraint, i.e., the ability of the system to make decisions on time scales appropriate to the expected behavior of the system. To investigate the computational time required for the proposed approach the average computation times were measured on two different CPUs. The results can be seen in Fig. D.7. Notice, that these timings are based on a relatively slow python implementation and are uniquely tied to the specific use-case presented in Section D.7.1 and Section D.7.2. As such, they should not be seen as the definitive timings that can be obtained utilizing the method, but rather as indicative of roughly what can be expected by the approach. Nevertheless, the timings given in Fig. D.7 would probably be too slow or jerky for most real-world robot applications. As should be clear from Fig. D.7 the most time-consuming part of the approach with the current implementation is the inference part of the *deliberate attention evaluations* step. As such, further optimization of this step would be needed to make the approach usable.

D.8 Discussion

The intention of the presented efforts was to implement general reflective mechanisms suitable for robotic applications with an outset in previous work. In Section D.7.1 and Section D.7.2 we demonstrate that the proposed method functionally improves upon our previous proposed probabilistic programming idiom and that it at least can perform as well as, if not better than, problem-specific methods. However, in Section D.7.3 it was concluded that the current implementation would probably be too slow and jerky for real-world robot applications. The approach presented within this paper is supposed to be generally applicable and reusable, making it hard to assess how much the current implementation should be improved to be applicable to real-world robot applications since this would of course depend on each specific use case. One way to asses this anyway could be by comparing it to Allan Newell's analysis of the time scales of human cognition [25]. This is reasonable because the ultimate end goal of our efforts is to make robots as capable as humans.

In a single cycle of *deliberate attention proposal*, *deliberate attention evaluation* and *affective response*, access to parts of cognition distal to the decision process have to have occurred multiple times in order to infer the motor buffer posteriors. This places the proposed approach somewhere above the "biological band" of Newell's analysis said to be on the order of ~ 10 ms. The next step up in Newell's analysis is to the "cognitive band" starting at the level of *deliberate acts* in the order of ~ 100 ms. However, the proposed approach does not merely comprise deliberation, i.e., choosing one known operation over other known operators by bringing available knowledge to bear, since operators are constructed for the to-be-produced response based on the proposed deliberate attention mechanisms. Therefore, the proposed approach also belongs somewhere above the time scales of *deliberate acts*. At the other end of the "cognitive band", we have *unit tasks* in the order of ~ 10 s. At the time scale of *unit tasks*, operations should be composed to deal with tasks. By design, the specific affective response presented in Section D.6.3 can only deliver simple responses in one decision cycle and not a plan of responses to solve complete tasks. This leaves us at the time scales of *elementary cognitive operations* or *immediate external cognitive behavior* at ~ 1 s. According to Newell's analysis, such elementary reactions often take $\sim 2-3$ s, however, with learning from experience, simplification, preparation, and carefully shaped anticipation, it can take less than $\sim 0,5$ s. By design, the specific affective response presented in Section D.6.3 can deliver simple responses within 1 to 3 full cycles of *deliberate attention proposal*, *deliberate attention evaluation* and *affective response*. With the timings in Fig. D.7a a response thus takes anywhere from ~ 1.7 s up to at most ~ 5.34 s in the case of two impasses. Thus, to arrive at the upper end of elementary reactions, the computational times of the current implementation would have to be improved with a factor of $\sim 2-3$. Obtaining such improvements does not seem implausible via code optimizations, however, it brings us nowhere near the lower end of $\sim 0,5$ s. This begs the question: can the proposed approach support the necessary machinery to learn from experience in order to deliver responses at the lower end of $\sim 0,5$ s?

In Section D.4 and Section D.5 we described the use of stochastic variational inference, as the basis for inferring a parametric approximation of the posterior over the motor buffer, $q_{\phi}^{\{i\}}\left(Z_{\text{Mb}}^{\{t-1\}^{+}}\right)$. It was assumed that this inference process would have to be done from scratch in each decision cycle. However, this need not necessarily be the case. Instead, we could make use of amortized variational inference [28–31]. Thus, instead of making use of a variational distribution with free parameters, ϕ, we would make use of a variational distribution with parameters determined by a parametric function, $\phi = f_{\phi}^{\{i\}}\left(Z_{\text{WM}\backslash\text{b}}^{\{t\}^{-}}, Z_{\text{LTM}}\right)$, e.g., a neural network. When new situations are encountered we would not necessarily gain much by doing so, however, over time this would in principle allow the system to generate proper responses to situations similar to those that the system has previously encountered, without performing any inference. Thereby, removing the need for the most time-consuming step in the decision cycle. Again, when considering the timings in Fig. D.7a, reducing the inference step to near zero, would bring the total time of a single decision cycle down to around ~ 500 ms with the current implementation. Now if it is possible to improve the other steps with a factor of $\sim 2-3$ via code optimizations, it would indeed seem plausible to achieve *immediate external cognitive behavior* in around $\sim 0,5$ s after an initial learning period.

Further optimization might be achieved by considering when to stop the underlying inference algorithm. In the current implementation, the underlying inference algorithm uses a fixed number of iterations that has to be pre-defined. It might not be necessary with the same number of iterations in all situations, and thus time could be saved if a more clever mechanism for deciding the number of iterations could be implemented.

With these additions and optimizations of the approach and its implementation, we believe that the approach will be applicable to real-world robot applications, and thereby contribute to the goal of constructing autonomous robots that can safely and naturally interact with humans while solving different abstractly and/or vaguely defined tasks. As such, these optimizations will be the focus of our future work.

Author Contributions: Conceptualization, M.R.D.; Methodology, M.R.D.; Software, M.R.D.; Validation, M.R.D.; Formal Analysis, M.R.D.; Investigation, M.R.D.; Writing – Original Draft, M.R.D.; Writing – Review & Editing, M.R.D., R.P., and T.B.; Visualization, M.R.D.; Supervision, R.P., and T.B.

Funding: This research received no external funding

Data Availability Statement: The software used for the simulations is available at [26]. The Github repository also contains configuration files with the

specific parameters and settings used for the experiments, as well as scripts to reproduce the two simulation experiments presented in this paper.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations The following abbreviations are used in this manuscript:

AKS	Active Knowledge Search
RGS	Reflective Goal Search

References

[1] K. A. Demir, G. Döven, and B. Sezen, "Industry 5.0 and human-robot co-working," *Procedia Computer Science*, vol. 158, pp. 688–695, 2019, 3rd WORLD CONFERENCE ON TECHNOLOGY, INNOVATION AND ENTREPRENEURSHIP"INDUSTRY 4.0 FOCUSED INNOVATION, TECHNOLOGY, ENTREPRENEURSHIP AND MANUFACTURE" June 21-23, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050919312748

[2] B. Fang, X. Guo, Z. Wang, Y. Li, M. Elhoseny, and X. Yuan, "Collaborative task assignment of interconnected, affective robots towards autonomous healthcare assistant," *Future Generation Computer Systems*, vol. 92, pp. 241–251, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X18316844

[3] F. Farid, M. Elkhodr, F. Sabrina, F. Ahamed, and E. Gide, "A smart biometric identity management framework for personalised iot and cloud computing-based healthcare services," *Sensors*, vol. 21, no. 2, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/2/552

[4] M. S. Kaiser, S. Al Mamun, M. Mahmud, and M. H. Tania, *Healthcare Robots to Combat COVID-19*. Singapore: Springer Singapore, 2021, pp. 83–97. [Online]. Available: https://doi.org/10.1007/978-981-15-9682-7_10

[5] M. R. Damgaard, R. Pedersen, and T. Bak, "Toward an idiomatic framework for cognitive robotics," *Patterns*, vol. 3, no. 7, p. 100533, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666389922001301

[6] J. E. Laird, C. Lebiere, and P. S. Rosenbloom, "A standard model of the mind: Toward a common computational framework across artificial intelligence, cognitive science, neuroscience, and robotics," *AI Magazine*, vol. 38, no. 4, pp. 13–26, Dec. 2017, article in a periodical. [Online].

Available: https://ojs.aaai.org/index.php/aimagazine/article/view/2744

[7] M. R. Damgaard, R. Pedersen, and T. Bak, "A probabilistic programming idiom for active knowledge search," in *2022 International Joint Conference on Neural Networks (IJCNN)*, July 2022, pp. 1–9.

[8] Y. Tashtoush, I. Haj-Mahmoud, O. Darwish, M. Maabreh, B. Alsinglawi, M. Elkhodr, and N. Alsaedi, "Enhancing robots navigation in internet of things indoor systems," *Computers*, vol. 10, no. 11, 2021. [Online]. Available: https://www.mdpi.com/2073-431X/10/11/153

[9] J. S. Grover, C. Liu, and K. Sycara, "Deadlock analysis and resolution for multi-robot systems," in *Algorithmic Foundations of Robotics XIV*, S. M. LaValle, M. Lin, T. Ojala, D. Shell, and J. Yu, Eds. Cham: Springer International Publishing, 2021, pp. 294–312. [Online]. Available: https://doi.org/10.1007/978-3-030-66723-8_18

[10] M. Boldrer, M. Andreetto, S. Divan, L. Palopoli, and D. Fontanelli, "Socially-aware reactive obstacle avoidance strategy based on limit cycle," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3251–3258, 2020. [Online]. Available: https://doi.org/10.1109/LRA.2020.2976302

[11] K. M. Krishna and P. K. Kalra, "Solving the local minima problem for a mobile robot by classification of spatio-temporal sensory sequences," *Journal of Robotic Systems*, vol. 17, no. 10, pp. 549–564, 2000. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/1097-4563%28200010%2917%3A10%3C549%3A%3AAID-ROB3%3E3.0.CO%3B2-%23

[12] P. K. Mohanty, A. A. Kodapurath, and R. K. Singh, "A hybrid artificial immune system for mobile robot navigation in unknown environments," *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*, vol. 44, no. 4, pp. 1619–1631, Dec 2020. [Online]. Available: https://doi.org/10.1007/s40998-020-00314-8

[13] C. Ordonez, E. G. Collins, M. F. Selekwa, and D. D. Dunlap, "The virtual wall approach to limit cycle avoidance for unmanned ground vehicles," *Robotics and Autonomous Systems*, vol. 56, no. 8, pp. 645–657, 2008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889007001741

[14] G. M. Sanchez and L. L. Giovanini, "Autonomous navigation with deadlock detection and avoidance," *Inteligencia Artificial*, vol. 17, no. 53 SPEC. ISS., p. 13 – 23, 2014.

[15] J. Alonso-Mora, J. A. DeCastro, V. Raman, D. Rus, and H. Kress-Gazit, "Reactive mission and motion planning with deadlock resolution avoiding dynamic obstacles," *Autonomous Robots*, vol. 42, no. 4, pp. 801–824, Apr 2018. [Online]. Available: https://doi.org/10.1007/s10514-017-9665-6

[16] J. E. Laird, *The Soar Cognitive Architecture*. The MIT Press, 2012.

[17] P. S. Rosenbloom, A. Demski, and V. Ustun, "The sigma cognitive architecture and system: Towards functionally elegant grand unification," *Journal of Artificial General Intelligence*, vol. 7, no. 1, pp. 1–103, 2017, article in a periodical. [Online]. Available: https://doi.org/10.1515/jagi-2016-0001

[18] A. Newell, G. R. Yost, J. E. Laird, P. S. Rosenbloom, and E. G. Altmann, "Formulating the problem space computational model," *Carnegie Mellon Computer Science : A 25-Year Commemorative*, pp. 255–293, 1991.

[19] D. V. Pynadath, P. S. Rosenbloom, S. C. Marsella, and L. Li, "Modeling two-player games in the sigma graphical cognitive architecture," in *Artificial General Intelligence*, K.-U. Kühnberger, S. Rudolph, and P. Wang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 98–108.

[20] K. Scherer, A. Schorr, and T. Johnstone, *Appraisal Processes in Emotion: Theory, Methods, Research*. Oup Usa, 2001.

[21] C. A. Smith and L. D. Kirby, "Putting appraisal in context: Toward a relational model of appraisal and emotion," *Cognition and Emotion*, vol. 23, no. 7, pp. 1352–1372, 2009. [Online]. Available: https://doi.org/10.1080/02699930902860386

[22] P. S. Rosenbloom, J. Gratch, and V. Ustun, "Towards emotion in sigma: From appraisal to attention," in *Artificial General Intelligence*, J. Bieger, B. Goertzel, and A. Potapov, Eds. Cham: Springer International Publishing, 2015, pp. 142–151. [Online]. Available: https://doi.org/10.1007%2F978-3-319-21365-1_15

[23] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. A. Szerlip, P. Horsfall, and N. D. Goodman, "Pyro: Deep universal probabilistic programming," *J. Mach. Learn. Res.*, vol. 20, pp. 28:1–28:6, 2019, article in a periodical. [Online]. Available: http://jmlr.org/papers/v20/18-403.html

[24] T. Li, D. Ho, C. Li, D. Zhu, C. Wang, and M. Q.-H. Meng, "Houseexpo: A large-scale 2d indoor layout dataset for learning-based algorithms on mobile robots," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5839–5846, article in proceedings.

[25] A. Newell, *Unified Theories of Cognition*. Harvard University Press, 1990.

[26] M. R. Damgaard, "Probmind," https://github.com/damgaardmr/probMind/tree/ec996e295575c384879b3d72cfc7e64b8085b9a5 (Accessed on 3. November 2022), 2022.

[27] M. R. Damgaard, R. Pedersen, and T. Bak, "Study of variational inference for flexible distributed probabilistic robotics," *Robotics*, vol. 11, no. 2, 2022. [Online]. Available: https://www.mdpi.com/2218-6581/11/2/38

[28] C. Zhang, J. Butepage, H. Kjellstrom, and S. Mandt, "Advances in variational inference," 2017. [Online]. Available: https://arxiv.org/abs/1711.05597

[29] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *Proceedings of the 31st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, E. P. Xing and T. Jebara, Eds., vol. 32, no. 2. Bejing, China: PMLR, 22–24 Jun 2014, pp. 1278–1286. [Online]. Available: https://proceedings.mlr.press/v32/rezende14.html

[30] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., Apr. 2014, pp. 1–14, article in proceedings. [Online]. Available: http://arxiv.org/abs/1312.6114

[31] R. Shu, H. H. Bui, S. Zhao, M. J. Kochenderfer, and S. Ermon, "Amortized inference regularization," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/1819932ff5cf474f4f19e7c7024640c2-Paper.pdf

Appendix D Affective Responses

```
Require: Deliberate Attention Evaluations of CA, SR $(Z_s^*)$, SRP $(Z_s^*)$, and SRE $(Z_s^*)$
 1: DAMS $\leftarrow \{\text{SR}(Z_s^*), \text{SRP}(Z_s^*), \text{SRE}(Z_s^*)\}$
 2: if $\nexists$DAM $\in$ DAMS : $p\left(z_\text{p}^{\{\text{DAM}\}} = 1\right) > P_{z_\text{p},\text{lim}}$ then   # No-change impasse
 3:     Run Algorithm D.2                                                           # Reflective Response
 4: else
 5:     repeat
 6:         DAM* $\leftarrow \arg\max_{\text{DAM}\in\text{DAMS}} p\left(z_\text{d}^{\{\text{DAM}\}} = 1\right)$   # pick the most desirable DAM
 7:         if $p\left(z_\text{d}^{\{\text{DAM*}\}} = 1\right) \geq p\left(z_\text{d}^{\{\text{SR}\}} = 1\right)$ then
 8:             # Only pick DAM* if it is more desirable than SR to avoid motoric
 9:             # actions driven mainly by the progress or information gain appraisals
10:             if $p\left(z_\text{Ac}^{\{\text{DAM*}\}} = 1\right) \geq P_{z_\text{c},\text{lim}}$ or $p\left(z_\text{Ac}^{\{\text{DAM*}\}} = 1\right) \geq p\left(z_\text{Ac}^{\{\text{CA}\}} = 1\right)$ then
11:                 effectuate $q_\phi^{\{DAM*\}}\left(Z_\text{Mb}^{\{t\}}\right)$
12:             end if
13:         end if
14:         DAMS $\leftarrow$ DAMS$\backslash$DAM*
15:     until DAMS $= \emptyset$
16:     effectuate $q_\phi^{\{\text{CA}\}}\left(Z_\text{Mb}^{\{t\}}\right)$                           # pick the constraint avoidance
                                                                                    strategy as a backup
17: end if
```

Algorithm D.1: Affective response for State reach

```
Require: Deliberate Attention Evaluations of CA, E, EP, and SR $(Z_s^*)$ if $Z_s^* \neq None$
 1: DAMS ← $\{E, EP\}$
 2: DAM* ← $\arg\max_{DAM \in DAMS} p\left(z_i^{\{DAM\}} = 1\right)$          # pick the DAM yielding most infomation
 3: if $\nexists DAM \in DAMS : \quad p\left(z_p^{\{DAM\}} = 1\right) > P_{z_p,\lim}$          # No-change impasse
      or $p\left(z_i^{\{DAM^*\}} = 1\right) \leq P_{z_i,\lim}$ then          # No information gain possible
 4:     $\mathbb{P}_{BT}$ ← set_path_to_backtrack()          # Generate the path to backtrack, $\mathbb{P}_{BT}$, from the state tree
 5:     Run Algorithm D.3          # Reflective Response
 6: else
 7:     repeat
 8:         DAM* ← $\arg\max_{DAM \in DAMS} p\left(z_d^{\{DAM\}} = 1\right)$          # pick the most desirable DAM
 9:         if $Z_s^* \neq None$
                and $p\left(z_i^{\{SR(Z_s^*)\}} = 1\right) \geq p\left(z_i^{\{DAM*\}} = 1\right) \cdot (1 - P_{z_i,\Delta})$          # StateReach gives nearly as much information
                and $p\left(z_p^{\{SR(Z_s^*)\}} = 1\right) > P_{z_p,\lim}$ then          # and sufficient progress
10:             DAM* ← SR $(Z_s^*)$          # If SR does not violate constraints it will be effectuated and Algorithm D.1 will be used in the next iteration.
11:         end if
12:         if $p\left(z_{Ac}^{\{DAM^*\}} = 1\right) \geq P_{z_c,\lim}$
                or $p\left(z_{Ac}^{\{DAM^*\}} = 1\right) \geq p\left(z_{Ac}^{\{CA\}} = 1\right)$ then          # No constraint violation?
13:             effectuate $q_\phi^{\{DAM*\}}\left(Z_{Mb}^{\{t\}}\right)$
14:         else
15:             DAMS ← DAMS\DAM*
16:         end if
17:     until DAMS $\neq \emptyset$
18:     effectuate $q_\phi^{\{CA\}}\left(Z_{Mb}^{\{t\}}\right)$          # pick the constraint avoidance strategy as a backup
19: end if
```

Algorithm D.2: Affective response for explore

```
Require: Deliberate Attention Evaluations of CA, E, EP, and $\mathrm{SR}(\mathbb{P}_{\mathrm{BT}}[\tau])$ for $\tau \in [1, ..., T_{\mathrm{BT}}]$
 1: $\mathrm{DAM}^* \leftarrow \arg\max_{\mathrm{DAM}\in\{\mathrm{E,EP}\}} p\left(z_{\mathrm{i}}^{\{\mathrm{DAM}\}} = 1\right)$        # pick the DAM yielding most information
 2: if $p\left(z_{\mathrm{i}}^{\{\mathrm{DAM}^*\}} = 1\right) > P_{z_{\mathrm{i}},\mathrm{lim}}$
      and $p\left(z_{\mathrm{p}}^{\{\mathrm{DAM}^*\}} = 1\right) > P_{z_p,\mathrm{lim}}$ then        # new information can be obtained
 3:     create_new_state_branch()        # add branch to state tree
 4: else
 5:     $\underline{\mathrm{DAMS}} \leftarrow \emptyset$
 6:     for $\mathrm{DAM}^* \in \bigcup_{\tau=0}^{T_{\mathrm{BT}}} \{\mathrm{SR}(\mathbb{P}[T_{\mathrm{BT}} - \tau])\}$ do        # SR for the first $T_{\mathrm{BT}}$ states in $\mathbb{P}_{\mathrm{BT}}$
 7:         $\underline{\mathrm{DAMS}} \leftarrow \mathrm{DAM}^* \cup \underline{\mathrm{DAMS}}$
 8:         if $p\left(z_{\mathrm{d}}^{\{\mathrm{DAM}^*\}} = 1\right) > P_{\mathrm{BT,min}}$ then        # state can be reached
 9:             if $p\left(z_{\mathrm{Ac}}^{\{\mathrm{DAM}^*\}} = 1\right) \geq P_{z_{\mathrm{c}},\mathrm{lim}}$        # without collision
                  or $p\left(z_{\mathrm{Ac}}^{\{\mathrm{DAM}^*\}} = 1\right) \geq p\left(z_{\mathrm{Ac}}^{\{\mathrm{CA}\}} = 1\right)$ then
10:                 $\mathbb{P}_{\mathrm{BT}} \leftarrow \mathbb{P}_{\mathrm{BT}} \setminus \underline{\mathrm{DAMS}}$        # remove the reachable states from the path currently being backtracked
11:                 if $\mathbb{P}_{\mathrm{BT}} = \emptyset$ then        # end of path has been reached
12:                     $\mathbb{P}_{\mathrm{BT}} \leftarrow$ set_path_to_backtrack()        # generate new path to backtrack
13:                 end if
14:                 effectuate $q_{\phi}^{\{DAM*\}}\left(Z_{\mathrm{Mb}}^{\{t\}}\right)$
15:             end if
16:         end if
17:     end for
18: end if
19: if $p\left(z_{\mathrm{Ac}}^{\{\mathrm{DAM}^*\}} = 1\right) \geq P_{z_{\mathrm{c}},\mathrm{lim}}$
      or $p\left(z_{\mathrm{Ac}}^{\{\mathrm{DAM}^*\}} = 1\right) \geq p\left(z_{\mathrm{Ac}}^{\{\mathrm{CA}\}} = 1\right)$ then        # DAM* is less risky
20:     effectuate $q_{\phi}^{\{DAM*\}}\left(Z_{\mathrm{Mb}}^{\{t\}}\right)$        # than collision avoidance
21: else
22:     effectuate $q_{\phi}^{\{CA\}}\left(Z_{\mathrm{Mb}}^{\{t\}}\right)$
23: end if
```

Algorithm D.3: Affective response for backtracking

ISSN (online): 2446-1628
ISBN (online): 978-87-7573-781-9

AALBORG UNIVERSITY PRESS