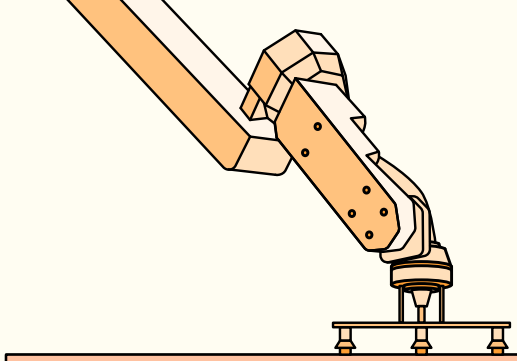
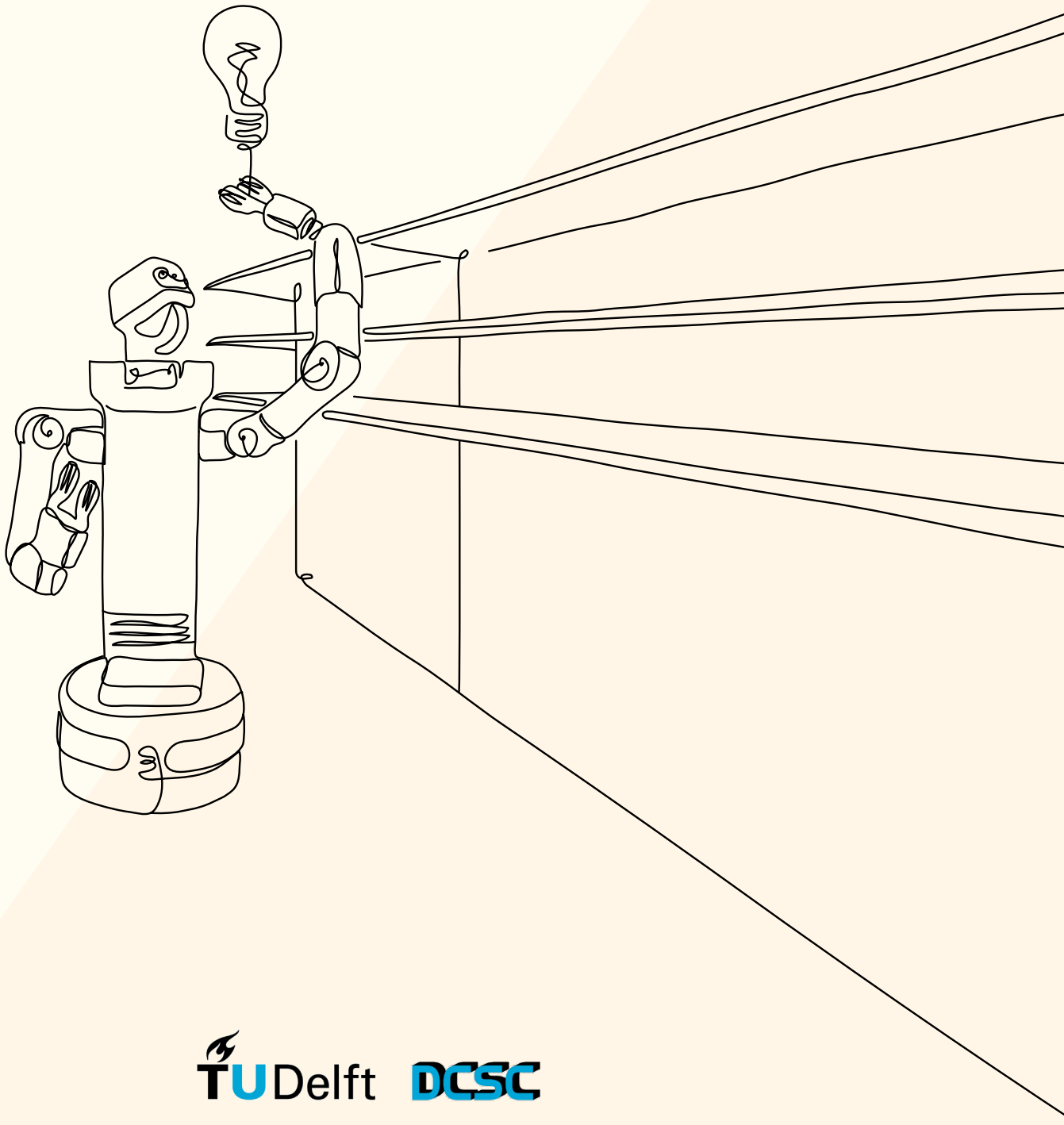


Knowledge-based Approach for Mobile Manipulation with Active Inference

Mohammed Mâachou



TU Delft DCSC

Acknowledgements

Guide us to the straight path – Al Fatiha.

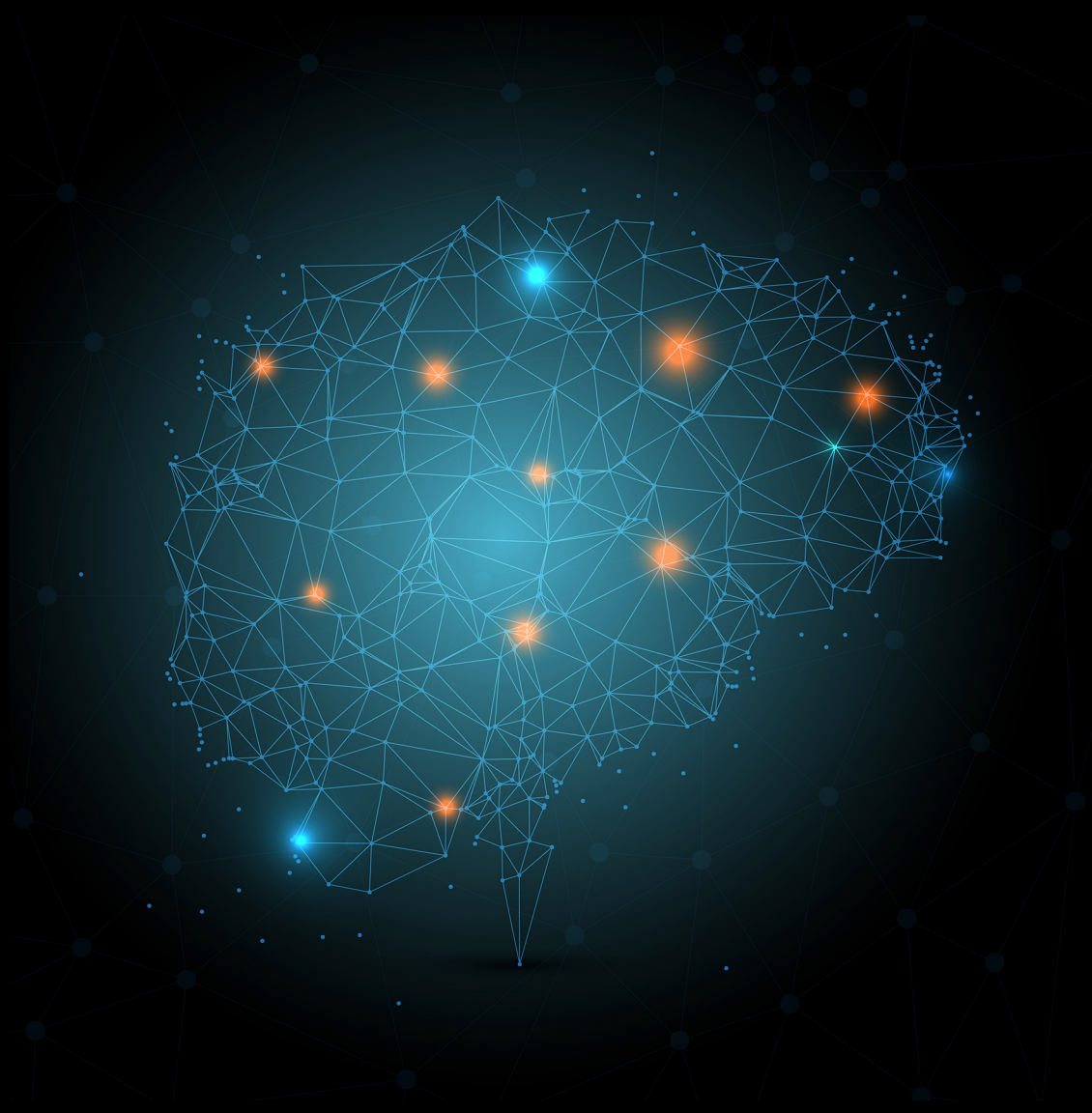
First, I would like to thank my supervisors Dr. Carlos Hernández Corbato, Dr. Riccardo Ferrari and Ir. Corrado Pezzato. Ir. Corrado, you helped me gain new insights, believed in me, and allowed the researcher in me to mature through critical yet accurate feedback. We had fruitful discussions together that will be missed. Next, I am truly grateful to Carlos for exposing me to new ideas worth exploring & investigating, always staying optimistic, and broadening my view on Robotics & Cognition. Also, I would like to thank Dr. Riccardo Ferrari for reminding me about the main message of my work. Your beneficial advice and broad perspective allowed me to focus on the message of my work to my audience.

Furthermore, I am thankful for Ruocha who was there for me during the difficult times of my thesis and a pandemic, actively listening to my struggles and providing great insights. You have revitalized my motivation and taught me to take pride in my work. Also, my friend Chadi, the machine father of TIAGo, provided great technical support and with whom I played many games of Quoridor together with my lab-mates. Next, I would like to thank my close friends Daniel, Pau, Sofian, Johnson, Nugah, Nikilesh, my sister Chaimae & her husband Arjan for giving me their perspectives on life through their companionship. Lastly, I want to thank my parents Noura & Mustapha who showed me the importance of a good work-life balance and their wisdom on coffee brewing.

Knowledge-based Approach for Mobile Manipulation with Active Inference

Mohammed Mâachou

Master thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

Knowledge-based Approach for Mobile Manipulation with Active Inference

Master thesis

Mohammed Mâachou

June 29, 2022

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Copyright ©
All rights reserved.

Abstract

Achieving human-like action planning requires profound reasoning and context-awareness capabilities. It is especially true for autonomous robotic mobile manipulation in dynamic environments. In the case of component failure, the autonomous robotic system requires reliable adaptation capabilities combined with a consistent understanding of the task, environment and the robot's capabilities for successful task completion. Recent research has shown that Active Inference, a unifying neuroscientific theory of the brain, has the potential to intrinsically handle substantial uncertainties in the system, resembling the adaptability of humans. These works, however, have the following limitations: (1) no distinction is made between actions with some commonality, capable of satisfying similar tasks, and (2) actions are assumed to be always feasible when preconditions are satisfied, regardless of their context. Given the situation, certain actions satisfying a task might not lead to task succession. This work proposes the AI for retail (Airet) framework, a novel extension of action planning through Active Inference for mobile manipulation. The Airet framework uses Bayesian networks and Ontological Reasoning to facilitate context-awareness in action planning through Active Inference. Reasoning on robot components, action-, manipulation- and environmental constraints is facilitated through a description-logic-based reasoner and an OWL-based ontology containing concepts relevant for action selection in a retail context. The capabilities of the Airet framework are demonstrated through the following cases (1) irrecoverable task & component failure prevention when dealing with ill-defined tasks, (2) Selection of the best action given the situation & the component capabilities through context-awareness (3) failure recovery & adaptation when dealing with component failure. Lastly, these situations are compared with research on reactive task planning through Active Inference without context-awareness. This thesis represents a leap forward from the current state-of-the-art in Active Inference for task planning in robotics, laying the foundations for further research in the direction of this thesis.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces the motivations behind this research, presenting three fundamental questions. Next, the main contributions of this research are highlighted. The chapter concludes with the outline of the thesis.

1-1 Research Motivation

The latest reveal of the humanoid robot in development by Tesla, characterised as their core product in development for the year 2022, is aligned with the surge of research and business interest in humanoid robots designed to overcome mundane tasks currently performed by humans [1]. Many of these mundane tasks require a deep understanding of the environment to come close to human-level adaptability and resourcefulness. This understanding of the environment includes handling unexpected situations and uncertainties. Retail is such a dynamic environment that is currently labour intensive and crucial for society, being dominated by mundane tasks. Many approaches exist attempting to provide this deep understanding for various domains, as can be found in surveys [7, 53]. Some data-driven learning approaches use artificial intelligence methods like neural networks and deep reinforcement learning to learn behaviour skills, often relying on data from demonstration and/or data from repeated attempts [71, 83, 48]. One notable approach uses deep reinforcement learning [42] which learns policies for mobile manipulation by training deep Q-functions through a variant of the Normalised Advantage function algorithm [43], speeding up the learning process concerning traditional deep-learning approaches. These approaches still require a significant amount of data, which increases with the increase of task complexity. Furthermore, in case of unforeseen events, retraining is often required to adapt to these anomalies for successful task completion or prevention of irrecoverable task failure. Another approach to obtaining a human-like understanding of manipulation is integrating semantic knowledge and reasoning to provide

[1] https://www.businessinsider.nl/elon-musk-says-teslas-humanoid-robot-is-the-most-important-product-its-working-on-and-could-eventually-outgrow-its-car-business/

deliberation. Several notable solutions are mentioned in surveys [65, 58, 69, 84, 23] [2]. Among the vast literature, promising frameworks are PMK [24, 23], KnowRob [82, 6], Skiros [76, 77] and MROS [17, 18, 46]. PMK facilitates task and motion planning knowledge by introducing concepts like manipulation constraints. Skiros allows for breaking down tasks into action primitives to facilitate the reuse of actions and construction of new actions. KnowRob allows for reusing episodic memories, which are recordings of actions in terms of controller inputs, together with semantic reasoning, to provide generic service tasks. MROS provides self-adaptation of the control architecture at run-time through meta-models encompassing the development and reconfiguration on top of a robotic agent control system.

Despite PMK & KnowRob being good frameworks for mobile manipulation, neither of these frameworks can provide reactive adaptation at run-time when dealing with contingencies, which is needed for the cognitive capability of decision-making & Choice. MROS & Skiros do allow for adaptation at run-time but lack the terminology for mobile manipulation in retail and do not contain standardised ontologies. Furthermore, MROS only triggers adaptation when failure has already occurred (being reactive); hence does not provide adaptation based on knowledge & constraints of future tasks. Skiros selects actions based on satisfaction of pre- and post-conditions, combining a hierarchical task network and a behaviour tree, which does not distinguish between actions for adaptation through context-awareness but instead selects them in a fixed manner. These limitations are especially important when dealing with the presence of different controller capabilities satisfying the same tasks (e.g. servo-electric gripper vs vacuum gripper). These limitations of adaptability (and decision-making & choice) are unwanted for a dynamic human-centric environment like retail. Inspiration from cognitive science can be taken to provide this reactive adaptation.

Some promising theories have been derived from cognitive science that might explain how biological or artificial agents can govern perception, action, planning, decision-making, and learning. One theory, in particular, stands out: the principle of Free energy & Active Inference by Karl Friston [31, 32, 37]. This theory is based on the assumption that agents try to satisfy expectations about sensations by minimising their free energy through observation and action-taking [21].

With this theory alone, the capabilities of artificial agents remain limited, let alone being close to that of humans, because it does not explicitly include common sense knowledge. Ontological reasoning frameworks can bring the aspect of knowledge closer to reasoning, and the combination of Active Inference and ontological reasoning can create fault recovery capabilities in agents. To conclude, the research motivations brought to the formulation of the following research question:

Can Ontological Reasoning be integrated into Active Inference for planning and executing tasks for mobile manipulation in a fault-tolerant manner?

Three sub-questions are derived, which, combined, can lead to a possible approach for answering the main research question. They are as follows:

[2] A collaboration effort of several creators of KRR frameworks for identification of major KRR frameworks is given in the Github page.

- ***Can the Active Inference structure be exploited to allow for context-based decision making?***

  Active Inference for robotics is as promising as it is intricate. To answer this question, A look at the state-of-the-art decision-making using the Active Inference algorithm is taken. For altering the output of the decision-making process through Active Inference, literature points towards implementing more complex generative models. This approach relies on learning through gaining information on the states and state transitions, leading to computational difficulties in the face of high-dimensional state-space and complex dynamics occurring in a human-centred environment, also known as the curse of dimensionality. Furthermore, a consequence of having to learn state transitions means that it does not allow for influencing decision-making in a real-time fashion under the influence of unexpected situations. To allow for a more reactive influence on decision-making, research shows that one can select a prior over plans, with a factor denoting how much the decision-making algorithm should trust this prior over the (Variational & Expected) free energy terms. This prior over plans is taken as a constant in most works on Active Inference, with the exception of some [81] which allow for learning the prior over plans by looking at how frequently certain plans are chosen (through gradient descent). These approaches also suffer from the drawbacks that come with learning not influencing decision-making reactively. Furthermore, they do not distinguish between plans based on contextual awareness but instead rely on the completeness and accuracy of the generative model parameters subjected to learning, mainly the state transition matrix and the likelihood of state-outcome matrix. *The gap this thesis fills is a method influencing the priors over plans with contextual knowledge and reasoning, populating the generative model, breaking this dependency on the blind learning of parameters of the generative model and the limitations that come with it.*

- ***How can ontological reasoning contribute to context-based decision making for task planning using Active Inference for mobile manipulation?***

  Literature is found on ontological reasoning for mobile manipulation, facilitating standardised terminology needed for context-awareness. This terminology is extended to facilitate context-aware decision-making with Active Inference and concepts needed for retail. Doing this simultaneously tackles limitations of ontological reasoning for mobile manipulation in terms of cognitive capability for deliberation being decision-making & choice, and drawbacks of learning targeted priors over plans for decision making facilitated by Active Inference.

- ***Will the integration of Active-inference with Ontological Reasoning frameworks provide high-level fault-recovery and adaptation?***

  Past works were identified, facilitating task-planning using Active Inference. The approach of Pezzato et al. [74] stood out by allowing prevention of failure by attempting to solve missing preconditions of tasks by prioritising tasks solving these preconditions. This method did not facilitate failure recovery through context-awareness and had drawbacks, one of which being alternative actions to achieve the same goal are chosen in a fixed order. It is far from optimal since the knowledge about the environment and interactions of the environment with the robot is missing from the action selection process. Hence this method cannot recover from controller failure in case of component redundancy, failure related to agents' capabilities and task specifications. In a retail

environment, one can think of the consequences being creating dangerous situations for humans, irrecoverable task failure and component failure. A framework is created by the author, extending the work of Pezzato et al. [74] on task planning for robotics to allow for failure recovery through context-awareness. This framework proposes a novel approach to context-aware decision-making using Active Inference. It is done by exploiting the Active Inference structure for facilitating decision-making and using and defining semantic knowledge and reasoning through description logics and Bayesian Networks.

Main Contributions

- ***Creation of ontological reasoning framework for mobile manipulation in retail, supporting Decision-making & choice***
- ***Adding context-aware decision-making in Active Inference by dynamically populating the prior over plans***
- ***Adding high-level fault recovery and adaptation for mobile manipulation with respect to previous work on Active Inference for Action Planning***

1-2 Thesis Outline

The document is organised as follows. Chapter 2 and 3 serve the purpose of making this thesis as self-contained as possible. These chapters give the necessary background on two main topics: Active Inference and Ontological Reasoning.

Chapter 2 introduces the background information necessary to understand the equations of Active Inference. This background information includes Bayesian Inference, Markov blankets, the definition of Surprisal and the mathematical definition of the Kullback-Leiber divergence. Next, the working principle of Active Inference and the meaning of the equations it is governed by are explained. It includes perception, belief update and action selection.

Chapter 3 provides the state-of-the-art in Ontological Reasoning, namely Ontological Reasoning frameworks for mobile manipulation, Ontological Reasoning logic and Ontological Reasoning for representing uncertainty through Bayesian networks. Next, relevant works using Active Inference for task planning in robotics are introduced with their benefits and shortcomings for planning and executing robotic tasks. A promising Task planning framework using Active Inference is further explained and analysed, taking into account situations that can occur in a retail store environment. Lastly, a lack of knowledge fueled action reasoning is identified as a missing element for more fault-tolerant and intelligent task planning using Active Inference.

In Chapter 4 an ontological reasoning framework for retail is devised, which allows for reasoning on mobile manipulation. The added concepts & properties of the ontology for retail are formalised, and the concepts from standardised ontologies for mobile manipulation and robotics are introduced in the ontology design. The reasoning logic supported by the created reasoner is shown, of which the main factors for reasoning are the environment, the robotic agent, the Action & Task.

In Chapter 5 several case studies are devised and implemented, demonstrating the capabilities of the devised Ontological reasoning framework in a retail environment. These case studies consist of Pick & Place tasks with the following cases.

1. **Controller failure**: In the presence of controller redundancy, controller failure occurs, and controller redundancy is present. The robot has to use context-awareness through ontological reasoning and Active Inference for failure recovery & adaptation, resulting in successful task completion.
2. **Different End-effectors**: In the presence of end-effectors with different capabilities, the robot has to reason on the most appropriate end-effector to use.
3. **Action failure guaranteed**: When dealing with an ill-defined task making it infeasible due to constraints of the object, environment or robotic agent, the robot has to use context-awareness to decline the task in a safe manner.

These cases are then tested against their counterparts, being the same cases with the Active Inference algorithm for task planning, as devised by Pezzato et al. [74], without ontological reasoning facilitating context-awareness.

Chapter 6 concludes the work presented in this paper and summarises the answers to the research questions previously posed. The author then includes guidelines for future research in the direction of this document, pointing out the main challenges and questions that remain unanswered.

Chapter 2

Background on Active Inference

This chapter gives an in-depth overview of the Active Inference principle, a free energy minimisation theory by Friston [31, 32]*, Initially, an introduction is given to the background information necessary to understand the equations of Active Inference. This background information includes Bayesian inference, Markov blankets, the definition of Surprisal and the mathematical definition of the Kullback-Leiber divergence. Next, the working principle of Active Inference and the meaning of the equations it is governed by are explained. It includes perception, belief update and action selection. The action selection equation is exploited in chapter 4 to bring cognitive awareness using ontological reasoning.*

2-1 Preliminaries

2-1-1 Background Information Free-energy Minimisation

In the following subsection, important concepts for understanding the concept of the Active Inference principle are briefly introduced. For a better understanding of these concepts, the author recommends to visit the following papers: [21, 34, 81, 8, 50].

1. **Markov Blanket:** Inference from observations of the outer world can be a costly process due to the significant amount of data available to sensors from the outer world. Much of this data can be filtered to relevant information based on the agents' internal model of the world and the desires that come with it. It is desired to obtain nodes with all the relevant information agents would like to perceive. For inference of a random variable, only a subset of variables is needed to be known which contains this useful information. This subset is called a Markov blanket. A minimal Markov blanket, also called a Markov boundary, has the following property: information is lost when one variable is dropped. The ideas of Markov blankets and Markov boundaries are formalised by Pearl [70]. A graphic depiction of a Markov blanket can be found in figure 2-1. In this figure, the Markov blanket for node {4} is the union of its parents {2,3,5}, its

children {8,9} and the children of the parents {6,7}. Node {1} and {4} are conditionally independent given nodes 2,3,5,6,7,8,9. In other words, when knowing variables around node {4}, the knowledge of node {1} does not provide extra information. In equation form, node 4 is given by $\{4\} = \{2,3,5\} \bigcup \{8,9\} \bigcup \{6,7\}$.

As for the conditional independence, this means that $P(\{4\}|\{1\},\{2,3,5,6,7,8,9\}) = P(\{4\}|\{2,3,5,6,7,8,9\})$.

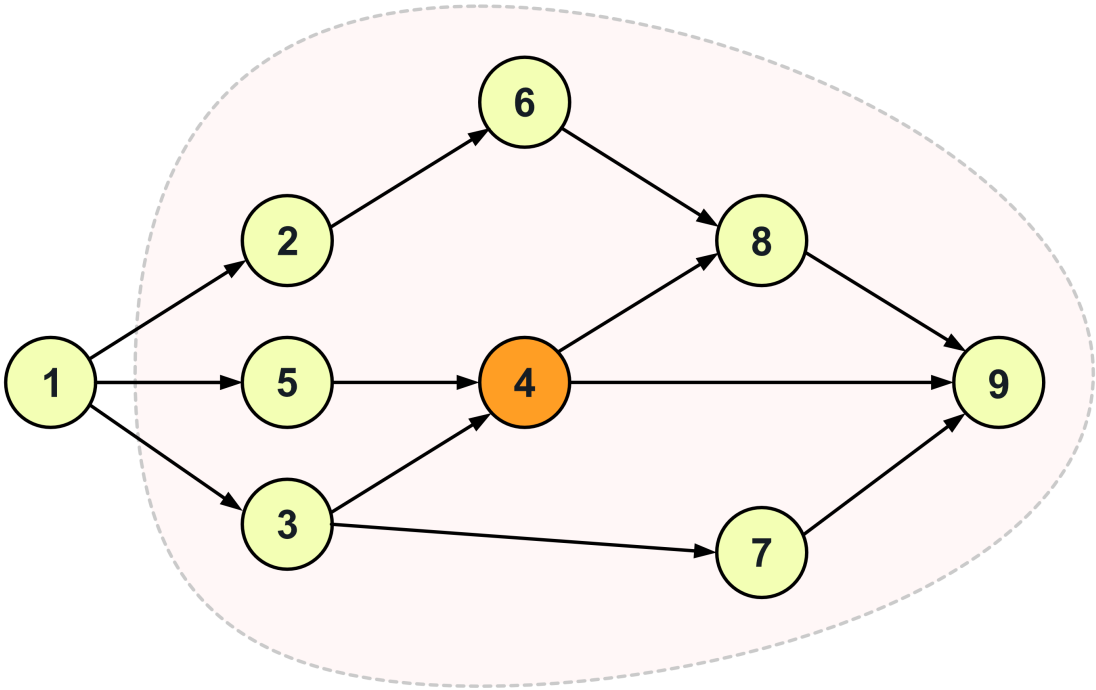


Figure 2-1: A graph depicting a Markov blanket with full conditionals

2. **Bayes' theorem**: The conditional probability $P(A \mid B)$ is defined as the probability of an event (A) occurring, given the knowledge that another event (B) has occurred. The joint probability $P(A,B)$ is the probability of events A and B happening. An example of conditional probability is the probability of a person being infected by the SARS-CoV-2 virus, given that one performed a SARS-CoV-2 test and the virus test results were negative (not having been infected). The joint probability would be the probability of testing negative on the SARS-CoV-2 test and having been infected. The joint probability is symmetric, which means that $P(A,B) = P(A \mid B)P(B) = P(B \mid A)P(A)$. The probability of events A and B occurring is the same as the probability of event A occurring given event B, times the probability of event B occurring. The same goes for the opposite when switching A and B. From this, one can derive Bayes' theorem by dividing by probability P(B), which results in the following:

$$P(A \mid B) = \frac{P(B \mid A)P(A))}{P(B)} \tag{2-1}$$

Similar Bayes theorem for $P(A \mid B)$ can be obtained by dividing by $P(A)$. The value $P(A \mid B)$ in this equation is known as the posterior probability. The denominator is

often referred to as the normalisation term, which ensures that $P(A \mid B)$ integrates to the value 1, i.e. for the continuous case $P(B) = \int P(B \mid A)P(A))dA$. Continuing on the SARS-CoV-2 test example, let the abbreviation COV mean the event that the SARS-CoV-2 infects one, and NEG being the event that one tests negative for the SARS-CoV-2. Let NOC stand for not having the SARS-CoV-2. The Bayesian posterior, determining the probability of having SARS-CoV-2 when one is tested negative, is given by the following formula:

$$fP(COV \mid NEG) = \frac{P(NEG \mid COV)P(COV)}{P(NEG)} \tag{2-2}$$

$$= \frac{P(NEG \mid COV)P(COV)}{P(NEG \mid COV)P(COV) + P(NEG \mid NOC)P(NOC)} \tag{2-3}$$

Notice that the probability of testing negative for SARS-CoV-2 is the sum of the probabilities for testing negative while having SARS-CoV-2 and while not having SARS-CoV-2.

3. **Kullbach-Leiber (KL) divergence:** As the name suggests, the Kullback-Leibler (KL) divergence D_{KL} was developed by Solomon Kullback and Richard Leibler in 1951 [52]. It is related to information entropy, also called Shannon entropy which is a measure of information, choice and uncertainty [79, 75]. In bits, Shannon's entropy can be used, for example, to quantify the amount of information obtained [75]. It is analogous to entropy as defined in thermodynamics. [88] The uncertainty for a set of possible states b_i with probability distribution $p(b_i)$ is given by its Shannon entropy:

$$H(p) = -\sum_i p(b_i) \ln p(b_i) \tag{2-4}$$

The Kullbach-Leiber (KL) divergence can be used to measure the relative entropy between two probability distributions. In the context of machine learning, it can give an indication of information gain for using probability distribution $Q(x)$ over $P(x)$. Furthermore, it is used for assessing image similarity, adapting discrete neural networks for model matching, free energy minimisation, and more [41, 94, 87]. It is the expectation of the logarithmic difference between probability distributions $Q(x)$ and $P(x)$. Formally, the discrete-time version of the KL-divergence is given as:

$$D_{KL}(Q(x)||P(x)) = \sum_x Q(x) \log \frac{Q(x)}{P(x)} \tag{2-5}$$

4. **Surprisal** or surprise can be comprehended as a measure of the level of the improbability of observation when relating a sensory state with an observation or sensory sample [36]. The time average of surprise is proportional to sensory entropy under ergodic assumptions (the assumption that a sample of a process is equally representative of statistical properties as the whole process when sampled for long enough) [50, 35].

5. **Agent:** The definition of an agent states that an agent is "*Any system that displays cognitive capacity, whether it is a human, a cognitive robot, or some other cognitive artificial entity*" [89].

6. **Generative model**: A representation of how the agent interacts with the world. It allows inference of external states representing the outer world and predicts future observations as well as consequences of actions. A simple generative model is given in equation 2-12. In literature, more complex generative models can be found [21, 34].

2-1-2 The Free Energy Principle & Active Inference

The Free Energy Principle (FEP) is based on the notion that organisms are unable to minimise Surprisal. Instead, they minimise free energy [50, 40, 33] which is an upper bound on surprise to ensure preferred outcomes are realised. Surprisal cannot be controlled directly by organisms. It is apparent for humans when being tickled compared to when tickling oneself. The latter does not cause surprise due to the cerebellum predicting our movement. The tickling action, hence, does not trigger laughter as a reaction of stimulation of the hypothalamus through tickling. Given continuous processes, organisms have an internal model of environmental states, referred to as the recognition density [8]. This internal model of environmental states is updated through approximating Bayesian Inference on the state of its environment, as is obtained from sensory observations. Furthermore, they rely on assumptions about how different environmental states shape sensory input in the form of a probability density function called generative density [8]. Acquisition of knowledge and adaptation to the agents' environment are methods of minimising surprise [36]. The benefits of free energy minimisation methods are that they can provide a mathematical foundation for adaptive behaviour and take learning and cognition into account. Furthermore, It can give an explanation of the biology behind the development and architecture of the brain.

The goal of the Free Energy Principle is to combine adaptive self-governing behaviour under the idea that minimising surprise is key for the survival of an agent.

The Active Inference principle

Active Inference extends the Free Energy Principle by assuming that taking actions, next to observing the surroundings to update the agents' beliefs, can lead to minimising uncertainty. Active Inference and the free energy minimisation principles have provided an understanding of how human brains function from both a physiological and a neuronal point of view [31, 38, 39].

In this work, discrete-time Active Inference is introduced for action selection and planning in discrete domains. Active Inference distinguishes itself from other Free energy minimisation methods by taking, besides perception, and action into account as a method of minimising free energy. Furthermore, the discrete-time formulation of Active Inference is chosen over continuous time. The discrete-time formulation lies closer to the application of Active Inference, partially due to observations of the external world by robotic agents being discrete.

Unlike reinforcement learning, Active Inference does not appeal to the concepts of reward, value and utility. Nor does it make use of the Bellman optimality equations [37]. The reward is seen as prior probabilities, and both exploration and exploitation are shaped into two components used to minimise the agents' free energy to maximise expected Bayesian model evidence [85]. These components are Variational Free Energy (VFE), which describes the level of alignment of an internal model of an agent and past sensory observations, i.e., how well an agent's beliefs describe the world and Expected Free Energy (EFE), which minimises

free energy by evaluating future actions with respect to prior preferences, i.e. which actions reduce the agents' uncertainty of the world in the future, as predicted now [21]. Furthermore, depending on the scenario, it can achieve similar results as reinforcement learning methods and can be scaled up to handle more complex machine learning problems [85, 20, 37].

Free energy is a bound on surprise, with the time average of surprise being entropy. Meaning the bound for entropy; hence the bound for surprise is obtained by a minimisation of free energy [50].

2-2 A Mathematical Introduction to Active Inference

Before arriving at a form of Active Inference, a few assumptions and notions need to be elaborated upon. The following assumptions are made before arriving at a formulation of Active Inference:

- The system is at a non-equilibrium steady state (NESS). This follows from the assumption that the agent can reach his preferred state after perturbation, and a steady-state probability density function must exist.
- Markov blankets: The world as the agent sees it is likely not as detailed as the world truly is. The agent can still, however, generalise this worldview by means of Bayesian inference from sensory observations to extract the relevant information an agent needs for survival. In other words, there is a boundary between the internal states of the agent and the external states belonging to the outside world. The relevant states for the interaction of the agent and the outside world can be seen in figure 2-2.

Model variable	**General definition**
S	Set of all possible (hidden) states, i.e. objects and events outside of the brain that cannot be known directly
s_τ	Hidden state at time τ.
$s_{1:t}$	sequence of hidden states $s_1, ..., s_t$
O	set of all possible outcomes
o_τ	Outcome at time τ
$o_{1:t}$	sequence of outcomes $o_1, ..., o_t$
T	Number of timestep in a trial of observation epochs under generative model
U	set of all possible actions
Π	set of all allowable plans, i.e. action sequences over time
π	plan or actions sequence indexed over time
Q	Approximate posterior distribution over latent variables of generative model
F, F_π	Variational free energy (VFE) and VFE conditioned over a plan
G	Expected free energy
Cat	Categorical (probability) distribution over finite set
$\boldsymbol{A}$ matrix	$P(o_\tau \mid s_\tau)$. A martix encoding beliefs about the relationship between hidden states and observable outcomes
$\boldsymbol{B}$ matrix	$P(s_{\tau+1} \mid s_\tau, \pi)$. A matrix encoding beliefs about how hidden states will evolve over time (transition probabilities)
C vector	$P(o_\tau)$. A vector encoding the degree to which some outcomes are preferred over others (prior expectations over outcomes).
D vector	$P(s_1)$. A vector encoding beliefs about a probability distribution over initial hidden states
E vector	$P(\pi)$. A prior probability distribution over plan, implemented as a vector
γ scalar	Encodes the precision estimate for the expected free energy over plan. It indicates how much trust should be put on the prior E with respect to G for plan selection, hence modulating the influence of G on plan selection.

Table 2-1: Nomenclature for understanding the equations governing the Active Inference principle

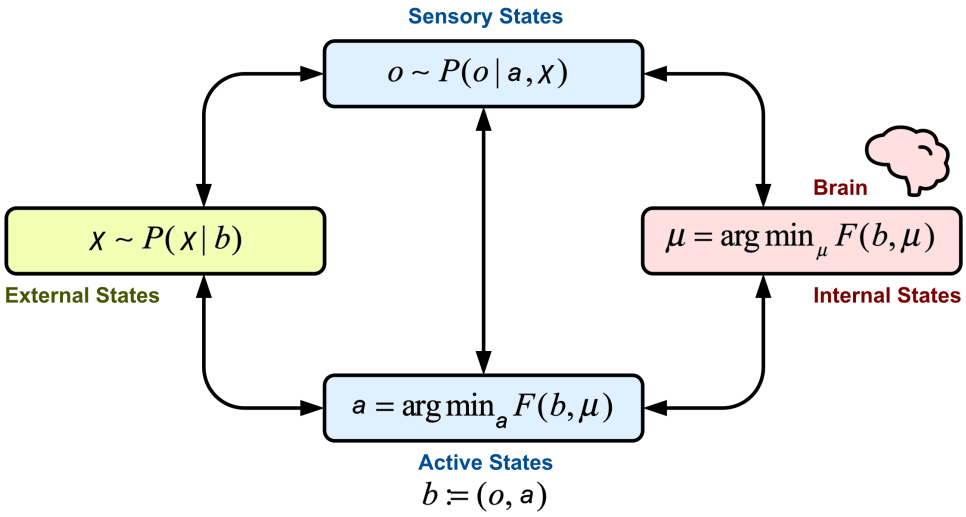


Figure 2-2: A graph depicting the states representing the interaction of the agent and the world, through Markov blankets, taken from [21]. Free energy minimisation states with ξ being external states representing the world, o being sensory states obtained from observations, a being active states which express how the agent influences the environment through acting, μ being internal states of the agent (not to be mistaken with hidden states s) and b being the blanket states between the world and the internal states of the agent.

Starting from Bayes theorem, equation 2-6, the agent desires to know the posterior distribution $P(s_{1:T}, \pi | o_{1:t})$, which encodes the probability of hidden states of an agent until time T, $S_{1:T}$, and action sequence π, given observed outcomes $o_{1:t}$. Computing this through the Bayes theorem is proven to be intractable, due to the denominator term depending on complex generative models ($P(o_{1:t}) = \sum_{\pi \in \Pi} \sum_{S_{1:T} \in \Pi} P(o_{1:t}, S_{1:T}, \pi)$) governing artificial and biological systems [30].

$$P(s_{1:T}, \pi | o_{1:t}) = \frac{P(o_{1:t} | s_{1:T}, \pi) P(S_{1:t} | \pi)}{P(o_{1:t})} \tag{2-6}$$

Instead, an option is to take an approximate prior distribution $Q(S_{1:T}, \pi)$ and optimise the distribution over latent causes. It is done to minimise the discrepancy between the approximate prior and the true prior. The KL divergence between these two distributions, together with the Bayes rule, shows an expression named Variational Free Energy which is able to be minimised in order for these distributions to be closer to each other. From this expression, one can derive that the difference between the approximate posterior beliefs and the generative model is always equal to or larger than a term that is referred to as surprise (negative probability of all outcome sequences). In mathematical form $-\log P(o_{1:t}) \leq F[Q(s_{1:T,\pi})]$.

By using the product rules of statistics, a well-known expression for the VFE is rearranged from the expression obtained by taking the KL divergence mentioned above to the expression in equation 2-7. This complexity term can be reasoned about as follows: A simple explanation for observable data Q, with few assumptions over the prior, equations 2-8, is a good justification for data requiring minimal change for updating prior to posterior beliefs. The accuracy term highlight how well the generative model fits the observed data.

$$F[Q(s_{1:T,\pi})] = \underbrace{D_{KL}[Q(S_{1:T},\pi) \parallel P(S_{1:T},\pi)]}_{Complexity} - \underbrace{\mathbb{E}_{Q(s_{1:T},\pi)}[\log P(o_{1:t}|S_{1:T},\pi)]}_{Accuracy} \tag{2-7}$$

$$Q(S_{1:T},\pi) = Q(\pi)\prod_{\tau=1}^{T} Q(S_\tau|\pi) \tag{2-8}$$

$$Q(S_\tau|\pi) = Cat(S_{\pi\tau}) \tag{2-9}$$

$$Q(\pi) = Cat(\pi) \tag{2-10}$$

$$\tag{2-11}$$

The generative model used in Active Inference can be described as partially observable Markov decision processes, see equation 2-12. Cat() in these equations stands for categorical or generalised Bernoulli distribution. Many generative models and approximate priors exist to take into account more parameters to be learned to represent the agent's beliefs better and minimise uncertainty. A derivation of Active Inference with a more complex generative model can be found in [34]. The only parameter learned in this generative model is the prior over plans $P(\pi)$ which comes in handy when integrating Active Inference with Ontological Reasoning frameworks.

$$P(o_{1:T}, s_{1:T}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{D}, \pi) = P(\pi)P(\boldsymbol{A})P(\boldsymbol{B})P(\boldsymbol{D})\prod_{\tau=1}^{T} P(S_\tau|S_{\tau-1},\pi)P(o_\tau|S_\tau) \tag{2-12}$$

$$P(o_\tau|S_\tau) = cat(A) \tag{2-13}$$

$$P(S_{\tau+1}|S_\tau,\pi) = Cat(B(u=\pi(t))) \tag{2-14}$$

$$P(s_1|s_0) = Cat(D) \tag{2-15}$$

$$P(o_\tau) = Cat(C) \tag{2-16}$$

$$P(\pi) = \sigma(\ln E - \gamma \cdot G(\pi)) \tag{2-17}$$

$$\tag{2-18}$$

Perception in Active Inference is equivalent to state estimation [34]. To infer states of the environment, an agent must minimise VFE with respect to $Q(S_{1:T}|\pi)$ for each plan π. The plan-specific free energy F_π can be expressed in terms of the priors as defined in the nomenclature from table 2-1, obtained from the generative model, and the hidden states conditioned on a plan, see equation 2-19.

$$F_\pi(\boldsymbol{s}_{\pi 1}, ..., \boldsymbol{s}_{\pi T}) = \sum_{\tau=1}^{T} \boldsymbol{s}_{\pi\tau}^T[\ln s_{\pi\tau} - \ln \boldsymbol{B}_{\pi_{\tau-1}}\boldsymbol{s}_{\pi\tau-1} - \ln A^T \boldsymbol{o_\tau}] \tag{2-19}$$

$$= \sum_{\tau=1}^{T} \boldsymbol{s}_{\pi\tau}^T \ln \boldsymbol{s}_{\pi\tau} - \sum_{\tau=1}^{T} s_{\pi\tau}^T \ln A^T \boldsymbol{o_\tau} - \boldsymbol{s}_{\pi 1}^T \ln \boldsymbol{D} - \sum_{t=2}^{T} \boldsymbol{s}_{\pi\tau}^T \ln B_{\pi_{\tau-1}}\boldsymbol{s}_{\pi\tau-1} \tag{2-20}$$

By taking the gradient of the VFE conditioned upon a plan, with respect to the hidden states, and setting this expression to zero, the posterior estimate of the state conditioned

by a plan can be found that minimised the VFE. This expression is given in equation 2-22. Note that σ is the softmax function and for $\tau = 1$ the first term becomes equal to D, i.e. $\ln B_{\pi\tau-1}s_{\pi\tau-1} = \ln D$.

$$s_{\tau\pi} = \sigma(\ln B_{\pi\tau-1}s_{\pi\tau-1} + \ln B_{\pi\tau} \cdot s_{\pi\tau+1} + \ln \boldsymbol{A}^T o_\tau) \tag{2-21}$$

$$s_{\pi(\tau=1)} = \sigma(\ln \boldsymbol{D} + \ln \boldsymbol{B}^T_{\pi\tau}s_{\pi\tau+1} + \ln \boldsymbol{A}^T \boldsymbol{o}_\tau) \tag{2-22}$$

$$s_{\pi(1<\tau<T)} = \sigma(\ln \boldsymbol{B}_{\pi\tau-1}s_{\pi\tau-1} + \ln \boldsymbol{B}^T_{\pi\tau}s_{\pi\tau+1} + \ln \boldsymbol{A}^T \boldsymbol{o}_\tau) \tag{2-23}$$

$$s_{\pi(\tau=T)} = \sigma(\ln \boldsymbol{B}_{\pi\tau-1}s_{\pi\tau-1} + \ln \boldsymbol{A}^T \boldsymbol{o}_\tau) \tag{2-24}$$

The Expected free energy formula is similar to the Variational free energy formula, with the difference of minimising the free energy of beliefs about the future states of the environment, i.e. matching $Q(s_\tau|\pi)$ to the preferred state $P(s_\tau|\pi)$with $\tau > 1$ instead of past and present states.

Minimisation of VFE ensures that the generative model is a good predictor of its environment. It allows the agent to accurately plan into the future by evaluating the Expected free energy to enable the agent to realise its preferences. When an agent reaches these preferences, the expected surprise of future states of being is minimised. This perception-action loop is depicted in figure 2-3.

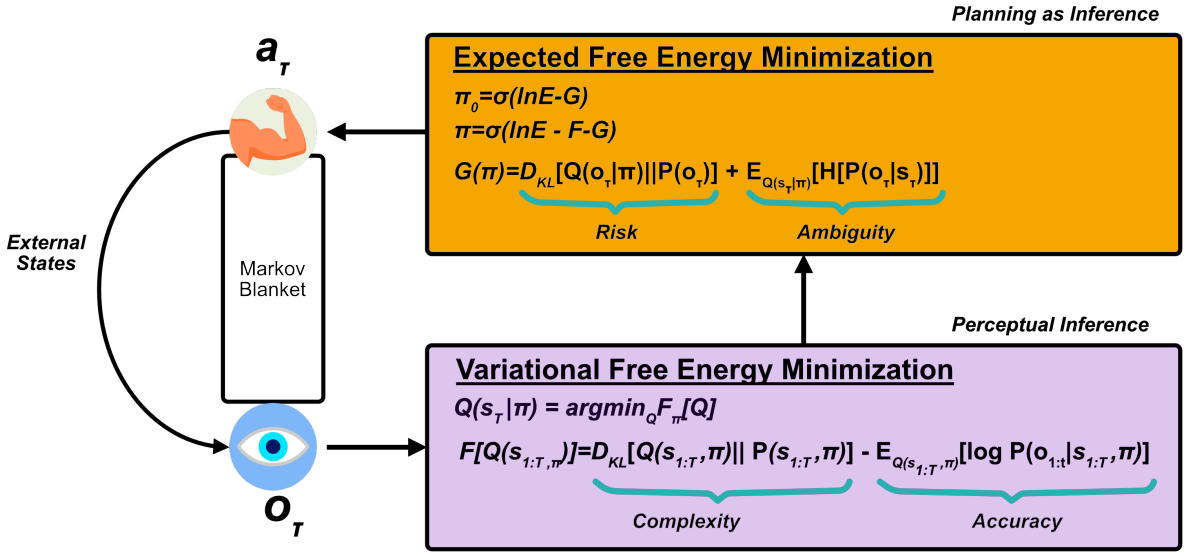


Figure 2-3: A graph depicting perception action loop of Active Inference. The variable definitions can be found in nomenclature table 2-1.

A well-known factorisation of the expected free energy is given in equation 2-25.

$$G(\pi) = \underbrace{D_{KL}[Q(o_\tau|\pi) \parallel P(o_\tau)]}_{Risk} + \underbrace{\mathbb{E}_{Q(s_\tau|\pi)}[H[P(o_\tau|s_\tau)]}_{Ambiguity} \tag{2-25}$$

with $H[P(o_\tau|s_\tau)] = \mathbb{E}_{P(o_\tau|s_\tau)}[-\ln P(o_\tau|s_\tau)]$.

Risk is the difference between predicted and apriori predictions in the future. Ambiguity is the uncertainty associated with future observations, given the states. The best plans are explorative and exploitative, meaning reducing risk and ambiguity, respectively.

Filling in the statistical terms from nomenclature, table 2-1, and simplifying, one obtains the following expression for expected free energy, given in equation 2-26.

$$G(\pi, \tau) = \boldsymbol{A} s_{\pi\tau}^T [\ln \boldsymbol{A} s_{\pi\tau} - \ln \boldsymbol{C}] - diag(\boldsymbol{A}^T \ln \boldsymbol{A})^T s_{\pi\tau} \tag{2-26}$$

Note that $\boldsymbol{A} s_{\pi\tau} = \boldsymbol{o}_{\pi\tau}$ after minimising KL divergence between observations expected given plan and preferred observations.

When taking the gradient of the VFE with respect to the plan this time, an expression for the update rule of possible plan distributions can be obtained. The result of this is the expression in equation 2-28. The initial plan distribution does not depend on the Variational Free energy since no past or present observations are present before time T=1, see equation 2-27.

$$\boldsymbol{\pi}_0 = \sigma(\ln E - \gamma G) \tag{2-27}$$

$$\boldsymbol{\pi} = \sigma(\ln E - F - \gamma G) \tag{2-28}$$

The E parameter denotes a prior over plans and can be seen as encoding 'habits' or preferences [81]. It tells which plans might be favoured over others, independent of current observations and hidden states. For a human, this might be picking up objects with the left arm compared to the right arm, assuming in this example that both arms result in the same outcome, and the task can be equally well satisfied by both arms.

The precision parameter γ , see nomenclature 2-1 gives a measure of trust of the effects of G on plan selection over the prior of plan E, see equation 2-28. A low value of γ indicates that the decision process is more influenced by 'habits' encoded in E than trust in the model beliefs generating desired outcomes. See the work of Friston et al. for the definition of habits and information on this precision parameter γ [81]. Action selection is determined from the most likely plan:

$$\pi^{max} = \max[\pi_1, \pi_2, ...\pi_p], \; a_\tau = \pi_{\tau=1}^{max} \tag{2-29}$$

2-2-1 Plan Selection Demonstrated

In the following example, the decision-making through active inference is demonstrated when taking into account prior over plans and when negating this prior. Assume an Active inference agent that operates in a simplified environment. This agent has two plans, π_1 & π_2, the first picking up an object and the second staying idle. The agent has a preference over observation encoded in matrix C, preferring picking up the object.

$$A = \begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}, \; C \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \; s_{\tau\pi 1} = \begin{bmatrix} 0.95 \\ 0.05 \end{bmatrix}, \; s_{\tau\pi 2} = \begin{bmatrix} 0.05 \\ 0.95 \end{bmatrix} \tag{2-30}$$

The observations under the two plans are:

$$\boldsymbol{o}_{\tau\pi 1} = As_{\tau\pi 1} = \begin{bmatrix} 0.86 \\ 0.14 \end{bmatrix},\ \boldsymbol{o}_{\tau\pi 2} = As_{\tau\pi 2} = \begin{bmatrix} 0.14 \\ 0.86 \end{bmatrix} \tag{2-31}$$

When computing the expected free energy for the plans, one gets:

$$G(\pi_1, \tau) = \boldsymbol{A}s^T_{\pi_1\tau}[\ln \boldsymbol{A}s_{\pi_1\tau} - \ln \boldsymbol{C}] - diag(\boldsymbol{A}^T \ln \boldsymbol{A})^T s_{\pi_1\tau} \tag{2-32}$$

$$\begin{bmatrix} 0.86 \\ 0.14 \end{bmatrix}^T [\ln \begin{bmatrix} 0.86 \\ 0.14 \end{bmatrix} - \ln \begin{bmatrix} 1 \\ 0 \end{bmatrix}] - diag\left(\begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}^T \ln(\begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}) \right)^T \begin{bmatrix} 0.95 \\ 0.05 \end{bmatrix} \approx 5.08 \tag{2-33}$$

Repeating this for the second plan gives us a value of 31.6037. Note that for computation, as is done before in literature [81], a value of 10^{-16} is taken to approximate zero, enabling computation since the logarithm of zero is minus infinity.

Assuming a value for the Variational Free energy $[F(\pi_1), F(\pi_2)]^T$ of $\begin{bmatrix} 1.83 \\ 1.83 \end{bmatrix}$, when one has the plan selection with a prior over plans having no preferences, $E_{np} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ the plan selection is as follows:

$$\boldsymbol{\pi_{np}} = \sigma(\ln E_{np} - F - \gamma G) = \sigma\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 5.08 \\ 31.6 \end{bmatrix} - \begin{bmatrix} 1.83 \\ 1.83 \end{bmatrix} \right) \approx \begin{bmatrix} 0.99 \\ 0.01 \end{bmatrix} \tag{2-34}$$

The most likely plan to satisfy our desires over observations C is the first plan, picking up the object. However, when suddenly a preference over plans is given, for example when the picking action cannot take place due to controller failure, the plans for picking is made infeasible by setting the prior over plans of this action to approximately zero. This prior over plans $E_p = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ will ensure that the robot stays idle, as follows:

$$\boldsymbol{\pi_p} = \sigma(\ln E_p - F - \gamma G) = \sigma\left(\begin{bmatrix} -36.8 \\ 0 \end{bmatrix} - \begin{bmatrix} 5.08 \\ 31.6 \end{bmatrix} - \begin{bmatrix} 1.83 \\ 1.83 \end{bmatrix} \right) \approx \begin{bmatrix} 0.01 \\ 0.99 \end{bmatrix} \tag{2-35}$$

One can see that the likely plan chosen is the second plan, staying idle.

2-3 Summary

Active Inference combines perception and action to minimise free energy, thus an agent's uncertainty about the external world. This principle can be used for learning, Action Planning and belief updating. Active Inference provides the cognitive capabilities: Planning & Problem Solving, Decision-making & Choice and Execution & Action. Decision-making through

Active Inference can be influenced by iteratively updating the prior over plans E, varying this parameter to allow for integrating context-awareness. The next chapter explains how this context-awareness can be created using ontological reasoning. In chapter 5, several case studies are devised showing how this context-awareness manifests into enhanced decision-making through combining Active Inference with Ontological reasoning.

Chapter 3

State of the Art

This chapter provides an in-depth overview and discussion of the most prominent works on ontological reasoning and Active inference. Ontological reasoning frameworks are introduced, as well as Bayesian network approaches facilitating reasoning with uncertainty. Furthermore, the state-of-the-art on Active inference for task planning and fault detection, isolation & recovery is presented, discussed and summarised. The main goal of this chapter is to find methods which could extend action-planning with Active Inference to include knowledge & reasoning for context-aware decision-making. This context-awareness should allow modelling and checking of the feasibility of actions.

3-1 Active Inference for Mobile Manipulation

3-1-1 Active Inference for Fault Detection, Isolation and Recovery

The works utilising fault detection, isolation and recovery focus primarily on detecting and recovering from a sensor failure. Fault recovery is then often performed by switching among different available fault-specific controllers [62]. Model-based methods for fault detection rely on mathematical models to generate residual signals to be compared to a threshold. The thresholds for fault detection used in these works utilising Active Inference can be grouped into using constant thresholds and probabilistic thresholds.

Pezzato et al. have developed a fault tolerance control (FTC) scheme based on Active Inference for robot manipulation with sensory faults [72]. Sensory prediction errors obtained from the difference between observed and expected sensory input, the latter obtained from the Active Inference algorithm, are used to generate residuals and thresholds for fault detection and isolation. Fault recovery is achieved by setting the precision matrix (or inverse covariants) of faulty sensors to zero. The advantage of this approach with respect to standard fault-tolerance approaches is that Active Inference intrinsically contains the signals needed for fault detection, isolation and recovery, hence removing the need for additional signals for monitoring or alternating controllers. This approach does have drawbacks, one being the

generation of false positives due to the adaptive nature of the Active Inference algorithm being biased towards reaching desired states. False positives can occur when goals change, leading to large sensory prediction errors which do not occur due to faulty sensors. Another drawback is the use of a conservative static threshold for fault detection. Baioumy, Pezzato et al. [4] have extended the works [73, 2] on adaptive control and state estimation using Active Inference and adapted the approach of [72] on an FTC scheme based on Active Inference. The main improvement of this approach is to introduce an unbiased Active Inference controller by reformulating the principle to allow the free energy to depend explicitly on the control actions and a probabilistic robust threshold (taking the Mahalobonis distance of residuals).

Baioumy et al. further introduced a fault-tolerant control scheme based on the unbiased active inference formulation of [4] for sensory faults in robotics manipulators using Active inference[3]. The main difference between the former approach and the latter is that the latter does not require a priori threshold definitions to trigger fault recovery. The authors achieve this by modelling the precision (inverse covariance) of each sensor in their system and by determining the probability of the sensors being healthy to be proportional to their respective precision. This allows for determining the degree to which sensors are faulty, compared to reasoning only on whether a sensor is faulty or not.

The discussed works can be utilised for low-level controller fault-detection and isolation. However, no work has been found on fault-tolerant control at the task level in the context of Active Inference. Nevertheless, past work could be used to trigger high-level fault recovery and adaptation useful for task-planning.

3-1-2 Active Inference for Task Planning and Execution

Kaplan & Friston have successfully simulated planning and goal-directed navigation of artificial agents using Active Inference in a maze [49]. One significant constraint of their approach is that prior beliefs had to be contextualised to generate feasible sub-goals.

Several deep Active Inference approaches exist for planning, which utilise a deep learning model, often a recurrent neural network, to learn parameters prior to generative models.

Using Active Inference for planning, computing the free energy for each possible plan deep into the future can lead to an explosion in a number of action sequences making it computationally costly [21].

Tschantz et al. [85] created a model of Active Inference that builds on previous deep Active Inference approaches[87, 10, 86, 60] to solve the curse of dimensionality and achieve goal-directed behaviour. The purpose of their work is to enable goal-directed behaviour for continuous control tasks and deal with high-dimensional state space and complex dynamics in the absence of reward observations. Free energy is minimised with respect to parameters of function approximators rather than parameters of the generative models themselves, using the cross entropy method [13].

Catal et al. developed a model that learns the state transition model and approximate likelihood matrix A of a system, see nomenclature table 2-1, plans by generating and evaluating trajectories of a search tree filled with parameters from a Monte-Carlo simulation [11]. The simulation learns how a car on a mountainous road can reach the top by adequately giving

gas. The likelihood matrix and transition model are approximated with values obtained from a connected neural network.

Matsumoto et Tani propose a goal-directed planning scheme inspired by predictive coding and Active Inference, investigating the problem of how agents can generate goal-directed plans based on learning using sensory-motor experiences [59]. This scheme employs a recurrent neural network to learn to extract the transition probability distribution of the latent state at each timestep as a prior.

Drawbacks planning frameworks All the approaches above require training schemes for acquiring an appropriate prior to the generative models. The work of Matsumoto and Tani [59] even requires supervisory learning to find attainable sub-goals. The difficulties with these approaches for robotic tasks are as follows: 1. Action preconditions cannot be taken into account, i.e., plans with conflicting actions are not addressed. 2. The methods are strongly limited to a certain context, hence requiring retraining for new tasks and environments. 3. They Are often computationally expensive, and planning happens solely offline 4. They do not have fault recovery mechanisms, i.e., after a failure, the reset button is pressed, which is hardly applicable to robotics in dynamic environments, and 5. They often suffer from a non-deterministic plan generation, meaning that optimal plans are hardly guaranteed.

The work of Kaplan and Friston [49] suffers from points 1, 2 and 4. A more promising and encompassing approach to planning for robotics tasks is given in the next section 3-1-2.

Active Inference for reactive task planning framework

Background on Behaviour Trees

A behaviour tree is a directed tree consisting of nodes and links used for plan execution in robotics. The nodes are directed from parent nodes to child nodes through edges. The root node is the only node that is not a child of a parent node. The leaf nodes are the child nodes and not parent nodes (i.e. have no child nodes). The execution of a behaviour tree begins with the root node sending a tick to its child node. A tick is an activating signal that allows for the execution of a child node. A child node returns a status to the parent node, which can be either running, success or failure. Running denotes that the child's execution is not yet finalised. Success is returned upon successfully achieving the child's goal. Failure is returned when this goal is not attained.

Nodes in a behaviour tree can fall under control flow nodes and execution nodes. Execution nodes contain actions or conditions.

- *Action nodes* perform actions and returns running, success or failure, the latter if the action cannot be executed. They are depicted as red boxes, see figure 3-1.
- *Condition nodes* do not affect the environment but check if a condition is met or not. They only return success or failure. They are depicted as orange circles, see figure 3-1

The return Control flow nodes consist of fallback, sequence, parallel or decorator nodes. The fallback and sequence nodes are the most important control flow nodes.

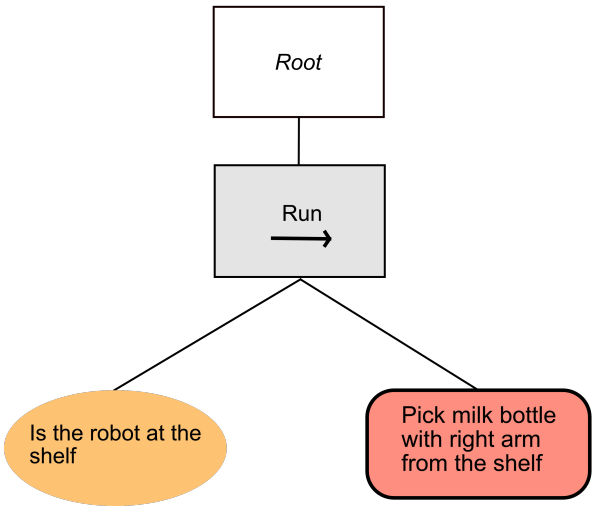


Figure 3-1: Simple behaviour tree example depicting an action, condition, sequence and root node. The action node (red square) executes a picking action dependent on a shelf location. The condition node specifies this location as a criterion for the picking action to be executed. If this condition is not met, the pick action is not executed, and failure is returned to the sequence node.

- *Fallback nodes* tick from left to right in order to find and execute the first child that does not fail. When this child returns success or running, the fallback node does not tick the next child. This node can be recognised by the question mark "?" symbol as depicted in (a) in figure 3-2.
- *Sequence nodes* are used to find and execute the first child that has not yet succeeded, also running from left to right. The sequence returns success only if all the children return success. If any child node fails, assuming there is more than one child node, then no tick is sent to the remaining child nodes. This node can be recognised by the horizontal right-pointing arrow "→" symbol as depicted in (b) in 3-2.

A more extensive explanation of behaviour tree formulations can be found in the work of Colledanchise et al. [15].

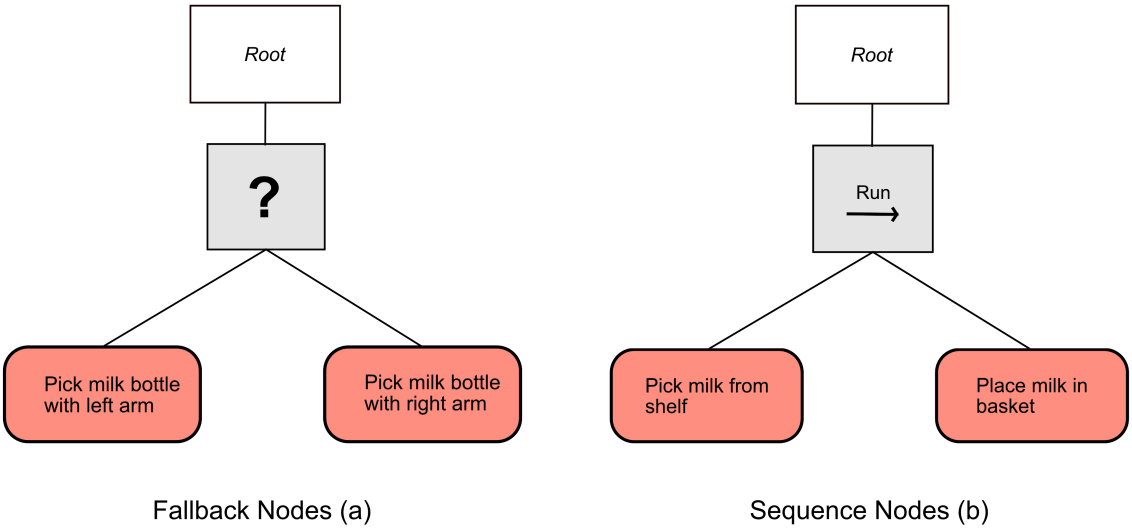


Figure 3-2: Simple behaviour trees depicting fallback nodes (a) and sequence nodess (b). (a) place action is only executed when pick action succeeds.(b) If picking with left arm fails, then picked with right arm will be attempted. When no failure occurs this node is skipped.

Method

Pezzato et al. showed how robotics tasks can be formulated as a free energy minimisation by combining the strength of behaviour trees and Active Inference for reactive action planning and execution [72]. The behaviour tree contains the flow of actions executed based on conditions and observations in a system. Behaviour trees suffer from the curse of maintainability, having to hard-code the conditions upon execution [16]. Pezzato et al.'s approach uses behaviour trees to contain desired states predefined offline, populating the prior preferences of outcomes C, see the section for nomenclature 2-1. Active Inference is used for online action selection and reporting the outcomes to populate further the prior defined in the behaviour tree. The extended Active Inference algorithm in Pezzato et al., [74] provides local reactivity to unforeseen situations while behaviour trees provide it on a global level. The results are that the algorithm designed adapts online to unforeseen situations, and takes action pre- and post-conditions into account while being context-independent; hence widely applicable to robotic tasks and contains fewer nodes than a sole behaviour tree approach. Lastly, deep plans are replaced with hierarchically composed shallow decision trees to handle the curse of dimensionality mentioned above.

Example Task plan: In figure 3-3 one can see an example task plan. The plan is fulfilled when the robotic agent moves to the location of the table, puts the milk bottle inside the basket, lifts the basket, moves the basket to the stocking shelf, and places it on the shelf. The behaviour tree for task T1 and T2 of figure 3-3 is given in figure 3-5. The task goals are formulated as desires which can be encoded in the vector C, see nomenclature 2-1.

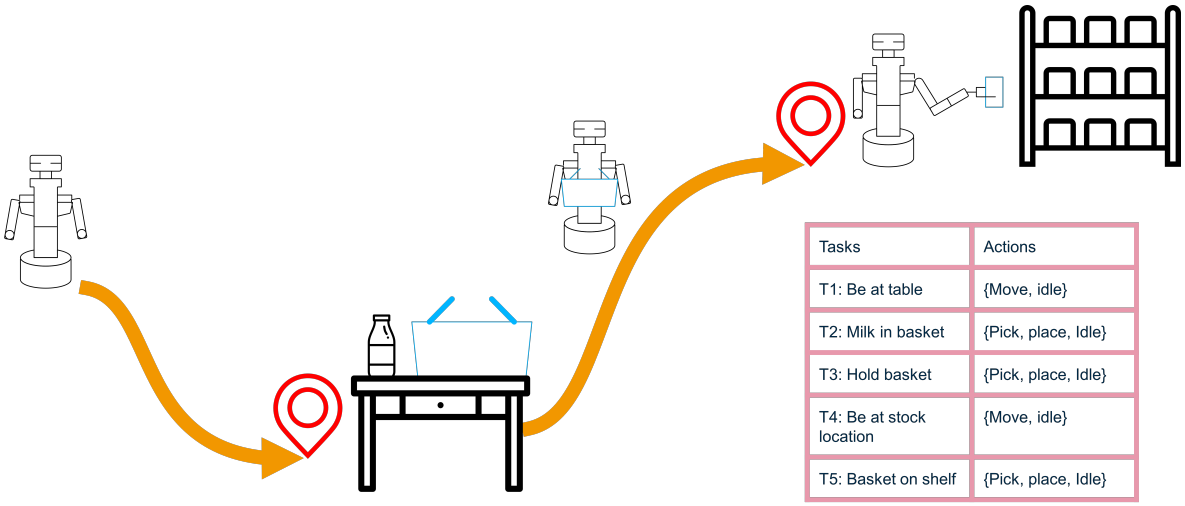


Figure 3-3: Example plan generated through the reactive action planning framework using Active Inference of Pezzato et a. [72]. The tasks are formulated as desire C and selected according to the plan created by an adapted behaviour tree.

The Active Inference algorithm chooses the task actions to satisfy these goals based on observations, model action templates (including pre- and post-conditions), prior preferences over observations C and inferred hidden states, see figure 3-4. Before a bottle of milk can be picked up by the robot, the preconditions of action pick need to be satisfied, see figure 3-5.

This online reactivity leads to the satisfaction of precondition isReachable(Milk). After these preconditions are satisfied, the pick task can resume, the behaviour tree ticks to the next task and the prior preference for this precondition is removed. The post-conditions indicate what a successful execution tells about the robot's belief regarding the hidden states. In this case, for reachability, the robot altered its bodily configuration to make the milk bottle within its reach. The matrix B encodes the probability of arriving at the next state given our current state and action taken currently. It could, for example, be the probability that the robot is holding a milk product, compared to not holding, after picking the milk product and the milk product being reachable.

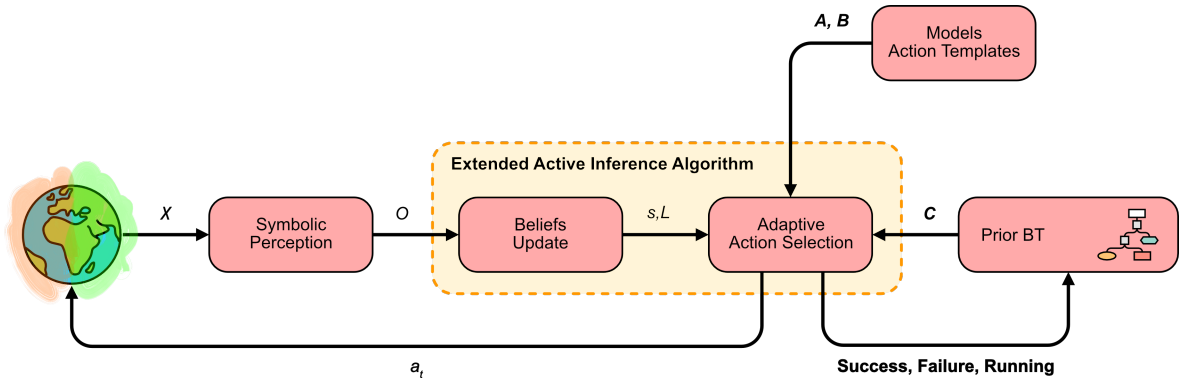


Figure 3-4: Active Inference for task planning scheme obtained from the work of Pezzato et al. [72]. The symbolic perception module translates percepts into observations understandable by the Active Inference algorithm. The model action templates include pre- and post-conditions and priors of the generative model. The behaviour tree includes prior preferences of observations, indicating the order of tasks to be fulfilled. The belief update of the Active Inference algorithm occurs through perception, and the adaptive action selection module specifies which actions to take based on pre- and post-conditions, the generative model, the hidden states inferred, and the task goal at hand. The variable definitions can be found in nomenclature table 2-1.

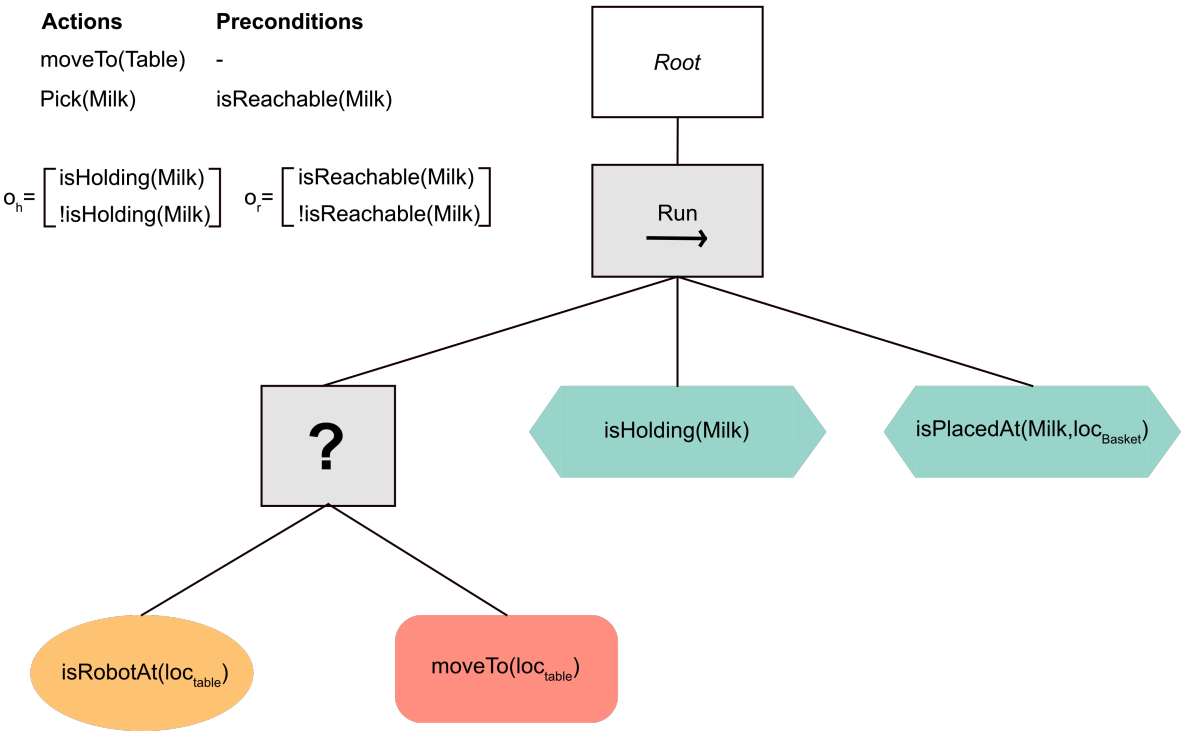


Figure 3-5: Example task plan satisfying tasks T1 & T2 of figure 3-3 through the reactive action planning framework using Active Inference of Pezzato et al. [74]. Actions and conditions are evaluated from left to right. The root node is where the execution starts. In orange, one can find a node containing a condition for execution. When this condition returns success, the blue action planning nodes are executed. When failure occurs, the red action block is executed. The preconditions stand for holding and being reachable, respectively.

Drawbacks Reactive task planning framework

This approach does have a few drawbacks.

- The behaviour trees containing preferences of states, the action models (including pre- and post-conditions) have to be hard-coded upon startup, which is a tedious task.
- Planning, including failure recovery, is as good as the accuracy of priors of the generative model.
- Alternative actions to achieve the same goal are chosen in a fixed order. It is far from optimal since knowledge about the environment and interactions with the robot is missing from the action selection process needed for selecting the right action for the task. For example, according to the model templates, in a pick-and-place task, a robotic agent using the extended Active Inference algorithm will try to pick up an object based on the first action that is thought to satisfy the task goal (desire). It can be either picking up with the left, right or both arms. Depending on the context, one of these actions might be more favourable than the other. E.g. with the desire to pick up a heavy object, the robot user has to encode in the action models that picking up with two arms is more desirable than with one. Object-specific and robotic actuators constraints are not taken into account or have been hard-coded by the user. In case of failure, however, one would desire to choose acting with the remaining arm that is functioning. Furthermore, when having different grippers, one would like to match the right gripper depending on the object properties and robot capabilities. These situations are depicted in figure 3-6. These preferences over plans can be encoded in vector E as a prior over plans, giving a probabilistic preference between [0,1] for each action. Currently, this value can only be hard-coded offline a posteriori after failure has occurred.

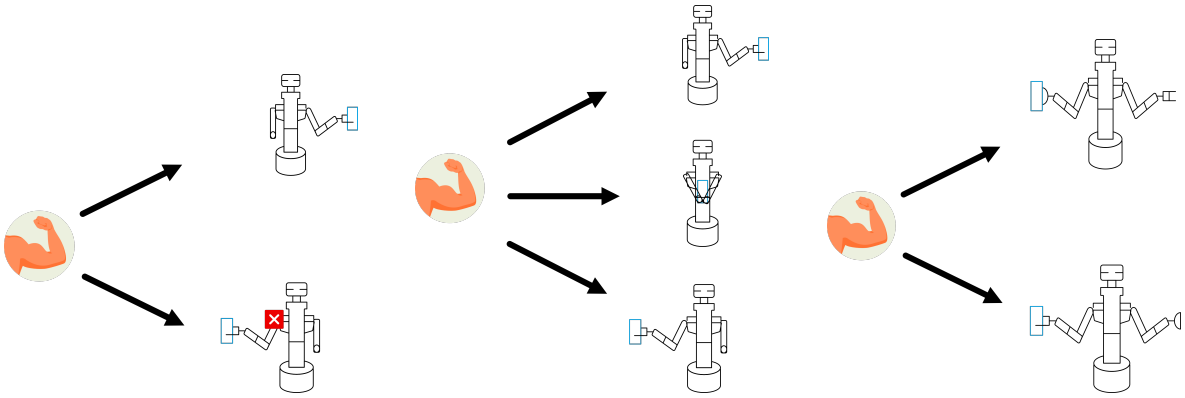


Figure 3-6: Three situations where certain actions might be more favorable over others depending on the situation. The left figure depicts the failure of the right arm. The middle figure shows different picking actions depending on the object size and location (using the left arm, both arms, or the right arm). The right figure shows different actions based on different gripper types. The upper right picture holds an object with a vacuum gripper while the lower right holds an object with a servo-electric gripper.

- Online adaptation of the global task plan does not consider controller failure. This failure ideally should lead to an adaptation of the global task. For example, lifting

heavy objects with two arms is no longer possible when one arm fails. Also, when having different grippers on each arm, particular objects requiring a specific gripper type cannot be picked anymore. The global task plan should be adapted online to exclude tasks that are not feasible anymore.

Besides the approaches mentioned above, no other methods combining Active Inference for planning have been found suitable for robotic tasks.

3-2 Knowledge Reasoning

In the previous section, the task-planning method by Pezzato et al. [74] using Active Inference in combination with Behaviour trees stood out by allowing for local reactivity to unforeseen situations. The drawback regarding the current state of the algorithm is choosing alternative actions to achieve the same goal (desired prior C) with a fixed order, and that action selection does not consider controller failure could negatively affect retail. The consequences of the first limitation include irrecoverable task failure, for example, attempting to pick an object with less suitable end-effectors, leading to dropping and damaging of the object, and controller failure, for example, through picking a heavy weighing object violating the payload of the end-effector, potentially damaging the controller. The second limitation for retail could have monetary and social consequences due to possible task plan adaptation not occurring. It could lead to, for example, essential products, like baby powder, not being stocked in time, resulting in loss of productivity and image of retailers among clients. By creating context-awareness, these drawbacks can be handled with a better understanding of the environment, the robotic agent's capabilities and components, and task & action constraints. The creation of context awareness is an intricate matter. Popular ways of creating context awareness for decades have been through knowledge representation and ontological reasoning. The following subsection provides relevant literature regarding knowledge representation and ontological reasoning, how this is utilised to create context awareness, and how this can be exploited for task planning in retail.

3-3 Ontological Reasoning Under Uncertainty

3-3-1 Knowledge Representation & Reasoning (KRR) Frameworks

Previous work by the author [58] have identified several promising KRR frameworks, including works not taken into account by previous surveys [65, 69, 84, 23]. These works are analysed, and novel characterisation of the most prominent KRR in terms of cognitive requirements for mobile manipulation is given in [58]. [1]. Of all the KRR frameworks identified and analysed, only the following frameworks were deemed suitable for knowledge representation & reasoning in a retail store, with possible integration with Active Inference. These are KnowRob, PMK, Skiros and Mros.

[1] A collaboration effort of several creators of KRR frameworks for identification of major KRR frameworks is given in the GitHub page

The cognitive capability of Decision-making & choice is the capability to represent different choices in an understandable matter to a robotic agent and choose between different alternatives. An example is choosing to pick an object over to place or stay idle when the robotic agent desires to hold an object. The robotic agent should understand through pre- and post-conditions the requirements and effects of taking this action. Robotic agents operating in retail are exposed to many different tasks with each requirement and desired course of action planning for task completion. Without this cognitive capability, the robotic agent jeopardises the safety and financial gain. Dependent on the situation, it could fail tasks, e.g. not being able to stock essential products which require a specific gripper type and, during controller failure, cause damage to the robot's component and environment. Several knowledge representation & reasoning frameworks were found to

Decision-making & choice is a cognitive capability that the following frameworks possess: OM, PMK, and Skiros.

- **The OM framework** runs on top of a system control module. Hence does not provide adaptation out of the box based on the agents' preferences of actions and task objectives. To take this into account, quality attributes have to be developed, encoding a model of the agents' preferences over plans, created as quality attributes. Currently, failure triggers adaptation. For mobile manipulation, in certain cases, it is desired to choose actions beyond component failure, actions that are more likely to achieve task succession. If this is to be implemented, quality attributes have to be designed carefully to drop in cases besides failure and to account for situations where certain actions are preferred over others.

  Furthermore, for adaptation to occur, killing and relaunching ROS nodes needs to happen, which is not desirable when dealing in a fast-paced, dynamic environment like a retail store. Lastly and most importantly, the current ontology does not contain TBOX and ABOX definitions relevant for mobile manipulation, nor concepts needed in a store environment. It also does not follow standardisation in its ontology. These concepts would have to be defined in a separate ontology, as well as actions, action constraints, robot constraints and components.

- **PMK** contains the TBOX and ABOX definitions needed to adapt actions based on reasoning on motion, perception and robot capabilities (sensors, grippers). Furthermore, it does follow standardisation in its ontology. It provides adequate mechanisms for decision-making & choice, which are centred around the ontology. However, when dealing with failure, it cannot adapt the task plan or the action selection after contingencies occur, causing controller failure during the execution phase.

- **Skiros** can adaptively match skills to the task at hand in an online fashion by using a behaviour tree combined with a hierarchical task network (HTN). The downside of this approach is that failure recovery is handled by the HTN, which mainly consists of switching between different skills satisfying different objectives based on their pre- and post-conditions. It does not include reactive adaptation due to unforeseen contingencies.

3-3-2 Bayesian Networks

Bayesian networks are a powerful and popular tool for efficient reasoning. Combined with ontologies, they can facilitate reasoning under uncertainty, which can influence decision-making through Active Inference. Many frameworks on ontological reasoning with Bayesian Networks exist, varying in domains, probabilistic concept representations and degrees of automation for constructing the Bayesian networks.

Early works on connecting Bayesian networks with ontologies are as follows: Helsper et al. [45] constructed Bayesian networks from ontologies by deriving classes and properties from the ontology. These classes are transformed into statistical variables. Conditional probabilities are not taken into account. Properties between the statistical variables are interlinked through arcs. Expert information or data is used to create these statistical variables. Furthermore, domain expert effort is required to check and remove the arcs for correct probabilistic independence.

Ding et al. created the BayesOWL framework [26, 25], which implemented probabilistic constructs in OWL that can be connected to individuals, classes and properties in an ontology for modelling and reasoning on the uncertainty of class membership of an individual. BayesOWL enhances OWL by adding OWL constructors like owl:intersectionOf, owl:unionOf, "owl:com- plementOf", "owl:equivalentClass", or "owl:disjointWith, in order to assign probability values to individual concepts, properties and conceptual relations. The probabilities are defined with the classes PriorProb, for prior probabilities, and CondProb, for conditional probabilities. BayesOWL contains a set of rules that are converted into probabilistic annotation, which can be inserted into a Bayesian network-directed acyclic graph (DAG), taking conditional probabilities into account. Furthermore, probability constraints are taken into account by using the iterative proportional fitting procedure (IPFP) during the construction of the conditional probability tables of the Bayesian Network. Two significant limitations of BayesOWL are (1) variables should be binaries, and (2) probabilities can contain a single prior variable only.

Yang et al. developed OntoBayes [93], which takes a similar approach to Ding et al., albeit Yang et al. focus on decision-making. The main differences between these works are (1) Yang et al. improved on BayesOWL by replacing the translation rules with a formal definition of "dependsOn", being **"A dependency is a pair X → Y, where each of X and Y is either a datatype property X.d or an object property s(X, Y). It reads as "X depends on Y"**. (2) OntoBayes models random variables as data or object properties. (3) OntoBayes allows multivalued random variable probability encodings in the ontology. (4) OntoBayes, opposite to BayesOWL, cannot model relationships between classes.

Costa et al. [19] created PR-OWL as a Bayesian ontology language for the semantic web. PR-OWL allows for reasoning on OWL concepts and definition of concepts subjected to probability by providing constructs for representing Multi-Entity Bayesian Networks. These constructs constitute a set of (sub-)classes, objects and data properties, unlike BayesOWL, which takes only (sub-classes) into account and OntoBayes, which takes only data Object properties into account. It does come with the cost of introducing more undecidability and intractability. In simple terms being intractable means a problem cannot be solved within polynomial-time and undecidable means there does not exist an effective method for deriving the correct answer (All true entailments cannot be found, all false entailments cannot be

refuted). Both intractability and undecidability have drastic effects on the computational effort it takes to find a solution to a logic problem through semantic reasoning. Both In PR-OWL, probabilistic concepts can coexist with non-probabilistic concepts.

PR-OWL 2 [9] solves some shortcomings of the first version. One is the lack of mapping between OWL properties and random variables used in PR-OWL [9, 56]. The other is a lack of compatibility with existing types prior available in OWL. The mapping issue is solved in PR-OWL 2 by having the properties of individuals correspond to random variables.

Some later works shift the focus towards creating domain-centred reasoning with ontologies using Bayesian networks. Li et al.[57] combine Bayesian networks with SWRL rules to reason on the evolution of emergency scenarios. Bayesian networks are used to perform conditional probability reasoning, in which the probability depends upon whether or not a condition specified by SWRL rules is satisfied. The construction of the Bayesian network relies on expert knowledge. Chang et al. [12] use ontologies and a Bayesian network to diagnose depression using the OntoBayes framework [93]. Depression symptoms are stored in the ontology, while the Bayesian network is used to determine the likelihood of having a symptom with the likelihood of having depression.

Numerous methods exist combining ontologies with Bayesian networks to create context awareness, often domain-specific. Some notable ones are by Gu et al. [44], and Ko et al. [51]. Gu et al. [44] combine Bayesian Networks and ontologies to deal with uncertainty in context. They defined two classes and two object properties, denoting conditional probabilities and dependencies between concepts. Ko et al. [51] combine Bayesian Networks and ontologies for context reasoning with uncertainty. Their contribution with respect to the work of Gu et al. [44] is the reuse of knowledge from uncertainty outputted from the Bayesian network to keep the benefit of reuse that is intrinsic with ontologies. With this framework, the output of the Bayesian network is used to find the correct configuration of a service. An example is when the Bayesian network outputs a high probability of the activity moving, then the probability of the robot being the service device is the largest.

Zheng et al. [95] create an ontology containing clinical practice guidelines with uncertainty, represented by a Bayesian network, into an ontology.

Wang et al. [92] propose using Bayesian networks with ontologies for giving personalised recommendations of tourist attractions. The ontology contains the user's profile as well as touristic attractions sourced from travel websites, while the Bayesian network estimates users' preference of attraction based on conditional probabilities dependent on age, travel motivation and occupation.

Methods not relying on modifying ontologies to include probabilistic concepts also exist, like the work of Fenz et al., where Bayesian networks are created based on concepts and individuals in an ontology, with relations between concepts relying on human input [28]. Furthermore, a conceptual scale is integrated into the ontology to estimate probabilities, consisting of terms like "high, medium low" for assessing security threats.

To sum up, no work has been found using Bayesian networks for enhanced decision-making using Active Inference.

The following challenges accompany the use of Bayesian networks with ontologies: (1) Identification of relevant variables for a considered domain (2) Identification of relationships between

identified variables (3) Creation of probability tables for each variable (in practice, conditional distributions in a Bayesian network model are unknown. Estimates can be obtained from expert knowledge or repeated experimental trials). (4) Construction of a measure representing the states of the identified variables. The benefit of using Bayesian networks with ontologies is to solely provide efficient reasoning with respect to formal logic-based approaches. Furthermore, ontologies with Bayesian networks can provide a method to model, account for and handle uncertainty, both semantically as well as situational awareness, given the uncertainty in observed data, to be used for decision making.

3-4 Summary

This chapter describes the most prominent works on ontological reasoning and Active Inference. Several works stood out for the purpose of using knowledge fueling action-planning with Active Inference for mobile manipulation in retail. The reactive action planning framework of Pezzato et al. [72] enables context-independent planning of robotic tasks with online action adaptation, including pre- and post-conditions. This method has drawbacks, one of which being alternative actions to achieve the same goal are chosen in a fixed order. It is far from optimal since knowledge about the environment and interactions of this environment with the robot is missing from the action selection process. Examples of knowledge are object, manipulation properties and robot capabilities. The planning and manipulation knowledge framework (PMK) contains necessary knowledge classes which can be used for reasoning on the robot's components and action constraints. Furthermore, the PMK ontology follows standardised ontologies CORA and SUMO. Doing so promotes the reuse of concepts and simplifies mapping between ontologies for different domains. However, this framework lacks the cognitive capability of Decision-making & Choice, not being able to reactively adapt actions given contingencies influencing the situation (e.g. controller failure). Using Bayesian networks constructed from the ontology provides an efficient and flexible approach to constructing preferences over actions dependent on probabilistic variables affecting the actions to be taken. It has the benefit over formal logic-based approaches in that uncertainty can be accounted for, which is aligned best with a dynamic environment being retail and Active Inference with beliefs governing this principle containing uncertainty. In the next chapter, a novel integration of Active inference & ontological reasoning with Bayesian networks is displayed for generating context-aware action planning. In chapter 5, several case studies demonstrate the strength of this framework.

Chapter 4

Active Inference for Retail (Airet)

This chapter introduces a novel framework for ontological reasoning with Task & Action Planning. This framework aims to create context-awareness for retail, leading to the best action chosen given the situation. This framework contains standardised concepts for the robotics domain & satisfies the most important cognitive capabilities for retail, taking reusability and extendability into account during the design process. This work exceeds other works in providing the capability of Decision-making & Choice by including reactive task adaptation satisfying missing preconditions, context-awareness resulting in an adaptation based on the component failure, and actions more likely resulting in task succession. This chapter starts with proposed novel methods by the author for integrating Active Inference with ontological reasoning. Next, the developed framework architecture is discussed, extending the behaviour tree to include reasoning nodes calling the reasoner module. Then, the Airet terminology is explained, and the terms selected for reuse from standardised ontologies and motion planning are highlighted. After, the functioning of the Airet reasoner is explained together with the reasoning capabilities and design freedom. Finally, an overview of cognitive capabilities and concepts crucial for autonomous robotic manipulation is given for the Airet framework.

4-1 Integration of Active Inference with KRR Frameworks

Integration of Active Inference and Knowledge Representation & Reasoning frameworks can facilitate the cognitive capability of Decision-making & Choice. Reasoning on constraints, the environment, and contingencies of the robot's capabilities and components and objects is crucial to making informed decisions and selecting the right action for the task. It is especially crucial in human-centred environments like retail, where safety and profit are crucial for daily functioning. Selecting actions without context information can lead to dangerous situations in retail and loss of company reputation. Examples are repeating actions with failing controllers, making unexpected movements, and not stocking essential products which require a specific action for their manipulation because of the product width and size. *The integration of ontological reasoning with Action Planning through active inference for robotics, as can be seen in figure 4-1, in this chapter, is a novel contribution to scientific literature.*

The author identified and proposed two main approaches in [58], which could be exploited to integrate Active Inference with knowledge obtained from KRR. It would bring Ontological Reasoning closer to Action Planning with Active Inference, as shown in figure 4-1. The first approach is data-centric, while the second approach is ontology-centric. They overlap in that the symbolic perception module translates percepts into observations understandable by the Active Inference algorithm. Also, the model action templates include pre- and post-conditions, and priors of the generative model. The behaviour tree includes prior preferences of observations, indicating the order of tasks to be fulfilled. The belief update of the Active Inference algorithm occurs through perception, and the adaptive action selection module specifies which actions to take based on pre- and post-conditions, the generative model, the hidden states inferred, and the task goal at hand. The variable definitions can be found in nomenclature table 2-1. These components are visualised as blocks in figures 4-2 and 4-3

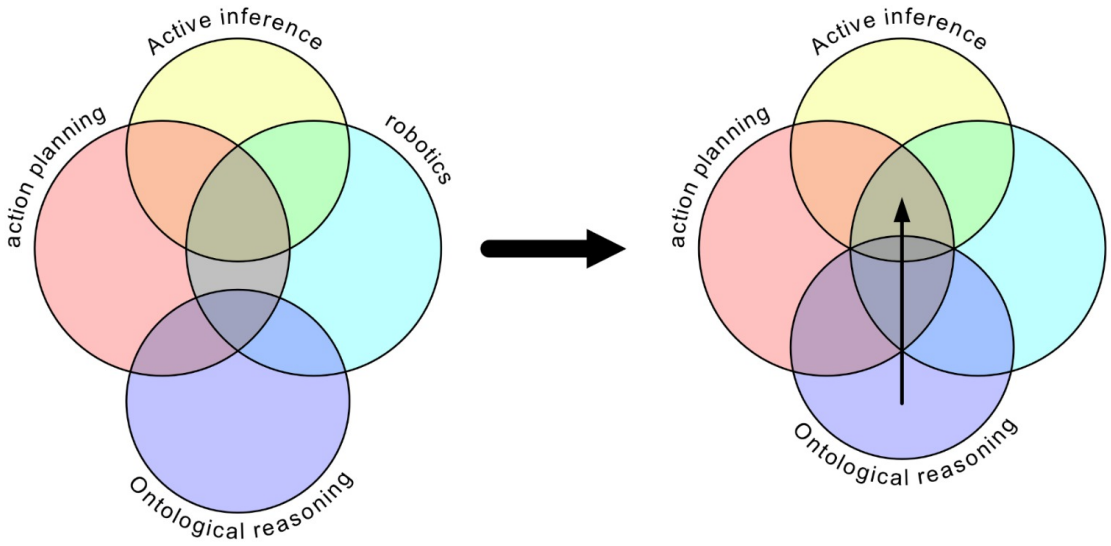


Figure 4-1: Figure demonstrating bringing Ontological Reasoning closer to Active Inference

Data-centred approaches rely on recorded data generated from sensor and actuator information based on the experiences of a robotic agent executing a specific task. A proposed Data centred integration by the author in [58] can be seen in figure 4-2. A pattern matching module has to be created to distinguish between situations occurring in the environment, taking actions into account. The encoded habits module is the database containing these recorded data generated from sensors and actuators, mapped to actions taken and knowledge extracted to identify the matching situation. The author identified data-centred methods to exploit the structure of these Ontological Reasoning frameworks for integration with Active Inference.

- **KnowRob** can be used to utilise the Recorded Episodic Memories (REM) containing data like positions, percepts, control signals and poses of successfully completed actions through real-world experiments and virtual simulations. Rem can be used to select actions based on similar situations encountered, giving a high value of E (close to value one) for actions stored in this database and lower values for those not stored. Furthermore, this method can be extended to record data of failed actions fueling the E

vector with probabilities of success and failure over the trials executed. The downside of this approach is that pattern matching tends to be complex in order for it to handle multiple situations, or it might fail to capture the essence of why actions fail, giving a lower value of a plan in E for actions that could have potentially provided better results in terms of reliability of success of an action. Another downside of this is that the KnowRob system often re-initialises REM upon startup; hence a method of saving REM is desired to have more data representing an adequate value for the fail-success probabilities.

- **Skiros** would need a ROS service module, which has to monitor the skill managers as well as the hierarchical task network used for planning to determine the probability of a primitive action and skill sequence to result in success in conjunction with the probability of successful exertion of each skill and primitive per task given. Also, another ROS service needs to be created to monitor the task goal and all the predicates and skills loaded with this task so that there could be a distinction between predicates applied concerning certain goals. An example to illustrate this is a predicate 'drive forward' might work when no obstacle is in front of the stocking shelves, but if there is a person in front of the shelves, this is impossible to execute. However, this is no reason to demotivate the robot to apply the skill drive forward, leading to inefficiencies.
- **PMK** framework can reason on the task at hand and the motion executed. An illustration in which the robot TIAGo++ is pouring a soda into a cup, provided by Diab et al. [24], shows that the robot adapts its motion based on the cup location and the location of itself [1]. Furthermore, it keeps track of similarities between current and previously encountered situations to determine if experimental knowledge is applicable to increase the success of applying the said skill. A service that communicates with the situational assessment module could be developed to determine which skills in the same situation are more likely to have failures for this situation and to reformalise a plan where these skills are less likely to be selected. The E matrix could then be filled with probabilities of actions in this situation that most often led to failure.
- **OM** framework, as is, encodes robotic behaviours through a predefined model of the system. This model can be used to encode information about the robotic agents' behaviour. For this to happen, quality attributes have to be defined concerning objectives that predict a drop in value before failure occurs. An option could be training a neural network to identify and predict drops in quality attributes based on the actions taken and quality attributes defined. Then a mapping encoding the beliefs over priors would have to be defined with a baseline defining a successful action, fuelling the E matrix discussed in chapter 2.

Ontology-centred approaches rely on predefined knowledge stored in an ontology and used by a reasoning module to extract relevant information dependent on the robot's capabilities, actions, tasks, object properties, and more. A proposed ontology-centred integration by the author can be seen in figure 4-3.

There is a clear winner in this ontology-centred approach which is the PMK framework. It is mainly due to the capability to reason on the robot, objects, actions, tasks, motions and situation at hand through semantic definitions in the ontology. It is due to its primary focus being

[1] https://www.youtube.com/watch?v=bTmWAkjC93c

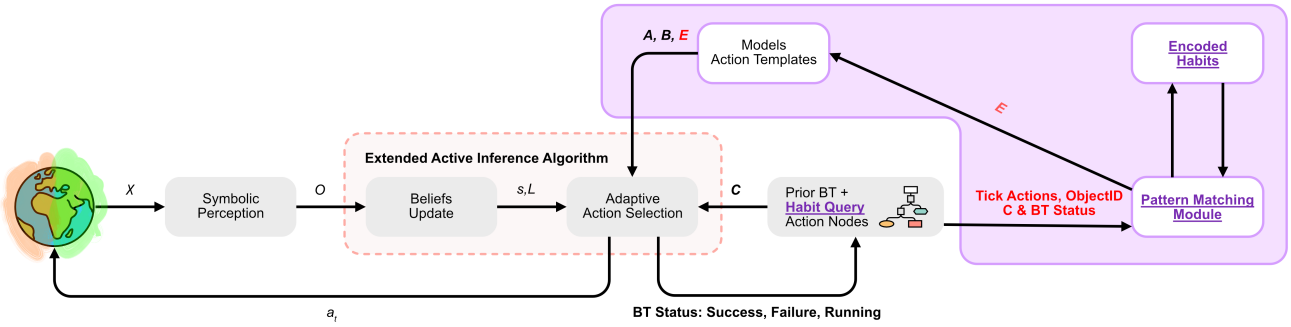


Figure 4-2: Possible data-centred approach developed by the author in [58], to integrating Active Inference for reactive task planning with improved decision-making based on pattern matching. The extensions over the reactive planning framework of 3-4 are depicted in purple. The reasoning action nodes would query the pattern matching module in the behaviour tree, also introduced in the extension. The pattern matching module would assess similarities between situations where similar actions have been taken and their outcome. Successful outcomes given a situation would be stored in the database, encoding habits as a probability of success or failure of these actions. Furthermore, it would provide the encoded prior over plan by querying the habits from the database. The behaviour tree is extended to include knowledge retrieval action nodes.

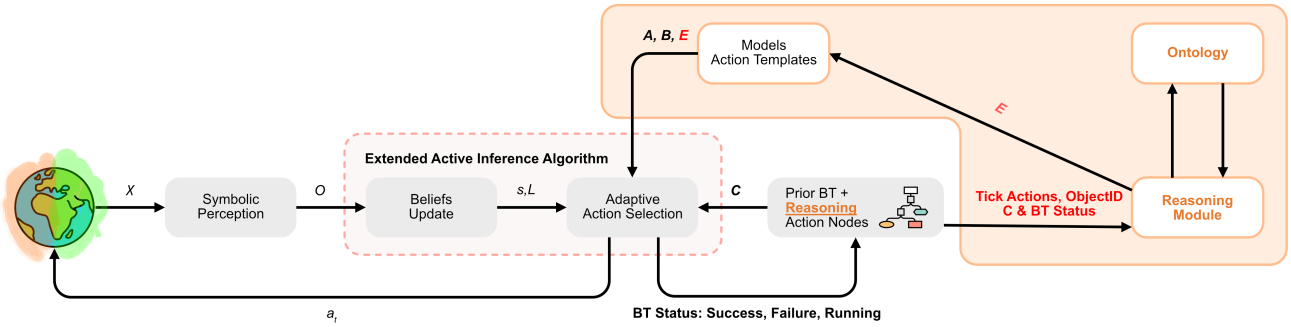


Figure 4-3: Possible ontology-centred approach developed by the author in [58], to integrating Active Inference for reactive task planning with improved decision-making based on Ontological Reasoning. The extensions over the reactive planning framework of 3-4 are depicted in purple. Reasoning on actions is possible through querying the ontology containing information on objects and their constraints, actions, motions, robotic agents and their components. The behaviour tree is extended to include reasoning actions.

to assist task and motion planning with knowledge. It facilitates defining constraints in the ontology for geometric reasoning, dynamic interaction, manipulation, and action constraints. Also, it satisfies more cognitive requirements than other frameworks, except KnowRob. Furthermore, its use case is service robotics which is most aligned with the use case for retail.

Another benefit of this approach is that it follows standardisation, which the closest opponent

KnowRob does not. Furthermore, KnowRob is known to have a bulky design suitable to infer properties of objects and their relations but over-engineered with details not needed for task planning in a retail environment where complexity in object definitions is not necessary. Furthermore, it lacks flexibility in planning and employing CRAM. No other method comes close to providing the reasoning capability on objects, object relations and motion constraints. A significant redesign of the ontology needs to take place for other methods to be applied for task planning using Active Inference.

Data-centred vs. Ontology-centred

Data-centred, unlike the ontology-centred approach, does not require expert information to populate ontologies and manual encoding of logic rules to select appropriate actions given a situation. The challenges of this approach lie primarily in designing and validating the pattern matching module. The decision-making using this approach might be subjected to learned features irrelevant to the context of action selection for mobile manipulation. The data-centric approaches require a vast amount of data and many trials before achieving reliable desired results of actions. It is to distinguish similar actions from those applied in different situations. Furthermore, expert knowledge in vision & machine learning is needed for the successful selection of actions. The ontology-centred approach is favoured over the data-centred approach since it requires the least amount of implementation efforts due to existing reasoning capabilities on constraints and the defined ontology, needing to populate the ontology by creating instances (ABOX) of defined classes in TBOX to fit a retail store.

In the next section, a proposed ontology-centred approach is developed utilising Bayesian Networks. Bayesian Networks require expert knowledge or data to estimate probabilities of events occurring. By keeping this approach ontology-centred, the ontology takes away most of the difficulty of situational awareness for the task at hand, the knowledge of the system and its components environment. Besides allowing for expert input, it enables more targeted data use or pattern learning.

The following section shall propose an ontological reasoning framework based on the PMK ontology, with a method of integrating Active Inference with Ontological Reasoning.

4-2 Framework Architecture

The context-based reasoning that is added to Action Planning with Active Inference consists of reasoning on the environment, the robot, its characteristics & its capabilities. This reasoning facilitates high-level failure recovery. Examples of characteristics are differences in reliable functioning between end-effectors or robotic arms. Examples of high-level failure recovery are choosing a functioning end-effector instead of a malfunctioning one, with the knowledge that a task is feasible. The cognitive architecture facilitating this context-awareness through the integration of ontological reasoning can be found in figure 4-4.

4-2-1 Airet Structure

The design of the Airet architecture, figure 4-4, takes deliberation into account. Deliberate acting refers to acting with sound reasoning justifiable with a goal in mind [47]. The necessities of deliberate acting are mentioned by Ingrand et. Ghallab, which are planning, acting,

observing, monitoring, goal reasoning, and learning [47]. Planning is handled through the Task, action and motion planning module. Acting is performed through the execution module, which responds to the present through appropriate instructions from the TAMP module. Monitoring is achieved mainly through the Active Inference algorithm, figuring out the discrepancies between the world model predictions of the agent and the real-world observations. Goal reasoning is achieved using the reasoning module with the TAMP module. It includes monitoring progress, constraints, failures and new opportunities. It decides when and how objectives should be updated, invalidated or discarded. Learning is facilitated using the reasoning module, ontology and Active Inference. These modules allow an agent to steer towards adaptation and, with that, improvement. Experiences are crucial for the learning process to achieve deliberate acting.

The perception module provides perceptions of the physical world to the robotic agent. Perceptions are interpreted by the reasoning module and the task, action and motion planning module (TAMP). The Airet reasoning module utilises semantic reasoning to select the appropriate actions a robot should take with the environment, task, and robotic component knowledge known to the robot. Furthermore, it checks for the feasibility of actions given a task defined in the task planning module. Percepts entering the reasoning module allow for identification of the object to be manipulated, reasoning on constraints from the environment and matching between modelled knowledge (belief of the robot about its environment) and the real-world environment. The semantic knowledge of the physical world, task and robot components capabilities are facilitated by the Airet Ontology. Percepts being fed into the TAMP module are interpreted in different manners for each element in the TAMP module. The motion planner uses these perceptions to attempt to find the existence and detail of the desired configurations for the required motion succession based on the robot controllers, obstacles detected and generated from sensor data manipulation and predefined obstacles. The task planning module consists of a behaviour tree specifying the user's predefined task goals. Percepts for the task planner annotate the success and failure of behaviours. An example of these goals is "be at a location x with pose y", with x and y being Boolean array structures containing the position and orientation of the robot. The action planner contains the Active Inference algorithm, which uses percepts to update the robotic agent's beliefs about the environment, select actions that lead to task succession and minimise uncertainty between the agent's beliefs and the percepts from the real world. The execution module obtains the target signals (e.g. joint commands) from the TAMP and contains the robot controllers, which execute this target signal and manipulate the physical world.

4-2-2 TAMP

This module comprises three components, task, action and motion planning, see figure 4-4. The task planner defines the task goals and the order in which tasks will take place. The action planner plans the actions that satisfy the goals defined by the task planner. The motion planner plans the configurations and the respective controller inputs to perform the action successfully. Furthermore, it also keeps track of the feasibility of executing actions. The task planner consists of an extended behaviour tree. The modification pertains to taking into account nodes with desired states of the robot to be reached through Action selection with Active Inference. *This is next to traditional goal states.* The Airet framework extends the behaviour tree to include reasoning action nodes, modelled as traditional action nodes

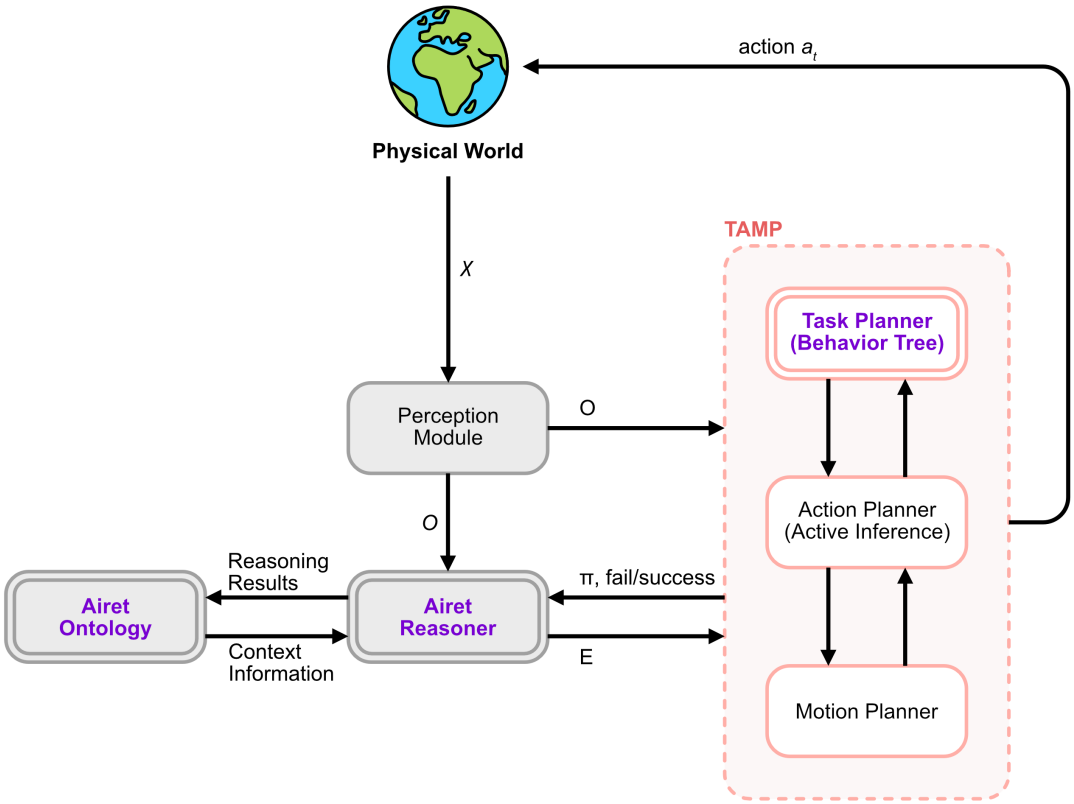


Figure 4-4: Figure displaying the cognitive architecture of the Airet framework. Ξ are the observations of the real environment, O are the observations translated to an understandable manner for the robotic agent. E is the prior over plan and a_t are the actions.

in the behaviour tree. Reasoning action nodes are created to reason on the feasibility of taking possible actions satisfying task goals modelled as Active Inference desired states. The result returned from this reasoning process directly influences the decision-making for Action Planning. Action Planning is performed by an extended Active Inference algorithm introduced in section 3-1-2. Most notably is the plan ranking equation $\boldsymbol{\pi} = \sigma(\ln E - F - \gamma G)$ which shows how the plan selection is influenced by a prior over plan E. In section 2-2-1 an example demonstrating the effects of this influence on plan selection is given. The results showed that despite the agent having a preference over observations resulting in favouring a certain plan, when this plan is infeasible, the prior over plans can change the decision-making into making this plan unfavourable. ***The work in this thesis focuses on influencing the prior over plans E through semantic reasoning and knowledge stored in the ontology.*** This prior over plans specifies which actions satisfying the same goal are more favourable, that is, after logic-based reasoning decides if they are at all possible, through context-awareness created by the Airet reasoner. More on this can be found in section 4-5

For motion planning, the Moveit framework [14] is used, incorporating the latest advances in motion planning, manipulation, 3D perception, kinematics, control and navigation.

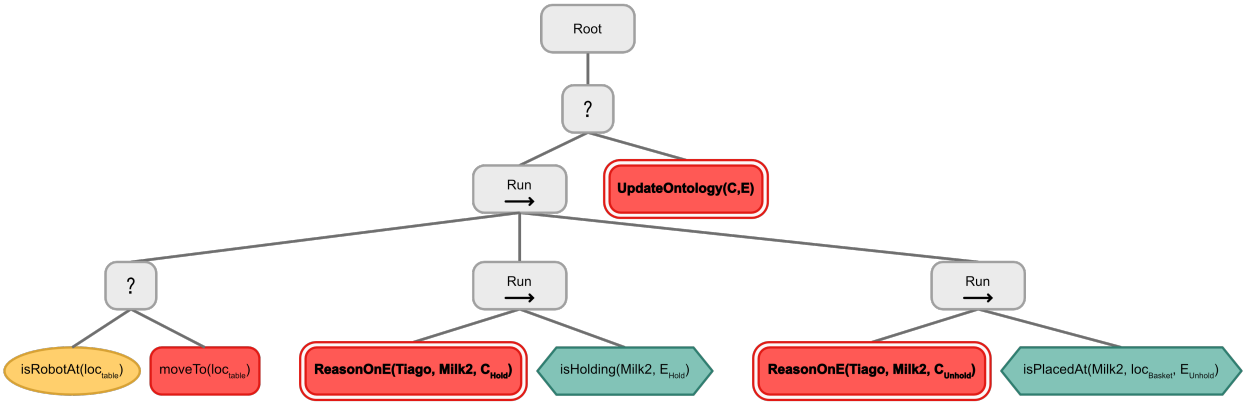


Figure 4-5: Example behaviour tree with extended reasoning nodes for task planning using Active Inference. The tasks are to move to a location, pick a pack of milk and place it in a basket. The sequence node is denoted with an arrow and the fallback node with a question mark. The action nodes (red square) execute a reasoning or execution action. The condition node (orange) specifies a criterion for executing the action. If this condition is not met, the pick action is not executed, and failure is returned to the sequence node. In blue, the encoded desires over outcome nodes are depicted, encoded tasks as desires for the robotic agent to achieve.

Where TIAGo++ is the robot type, Tiago1 is the robot individual, pickObj is the ask to be satisfied, and Hagelslag1 is the object individual to be held by the robot end-effector, as defined in the Airet ontology. The reasoner node is compatible with reasoning on Active Inference desired states and with standard behaviour tree action nodes.

4-3 Influencing Run-time Decision-making By Varying E

As was explained in chapter 2, the equation for plan selection $\boldsymbol{\pi} = \sigma(\ln E - F - \gamma G)$ governs decision-making through Active Inference by ranking actions with respect to their minimisation of the Expected Free Energy. Three factors influence decision-making, namely the prior over plans E, the Variational Free Energy F and Expected Free Energy G. The Airet reasoner influences the decision-making by specifying and varying vector E through context-awareness and feasibility of actions generated using the ontology and the Airet reasoner. To demonstrate the effects of the Airet reasoner on the decision-making through Active Inference, a snapshot is taken of the decision-making process right before selecting an action. This means that enough observations have passed for the perception loop of Active Inference to accurately depict the world, i.e. the robot's beliefs on location and on not holding objects match observations. In this snapshot of decision-making, F and G are constant for this timestep. One can better understand the plan selection when looking at the terms $\ln E - F - \gamma G$.

Two figures demonstrating the action selection are given in 4-6 and 4-7. Only the parameter E is varied with a range of zero to one for each action that that can successfully complete a picking task ($E_{Pick\ Left}$ & $E_{Pick\ Right}$), in this snapshot. By incrementally varying the $E_{Pick\ Left}$ in an opposite manner to $E_{Pick\ Right}$, one can see that there is a turning point where one of these actions is preferred over the other action, as can be seen in figure 4-6.

By varying the $E_{Pick\ Left}$ & $E_{Pick\ Right}$ in the same direction, one can see that if there are small differences between both of these values, the value with a higher prior over plans E is preferred over the other, given that the E values of the other actions are lower than both $E_{Pick\ Left}$ & $E_{Pick\ Right}$.

In figure 4-7 one can find that E values for staying idle and picking are equal to 0.37, which is displayed as a purple asterisk. It means that with a prior over plans below 37% of action success, the action staying idle is preferred. Above this value, the action picking with the right arm is slightly preferred over picking with the left arm (by design). This threshold can be varied by influencing the factor γ in equation 2-28, which indicates how much priors over plans should be preferred with respect to the Expected free energy term. In both figures, 4-6 and 4-7, one can see that the place actions are always lower than action idle, which is programmed by design since place actions alone cannot satisfy the desired state to be holding (an object). Furthermore, place actions can alter the environment, which is not the case for staying idle. In situations where both actuators fail or contextual constraints are violated, it is best to reject this task (by choosing action: stay idle) to prevent controller damage and unrecoverable task failure. In figure 4-6 one can see that at approximately 50, the selection of plans picking with the right end-effector is as desired as picking with the left end-effector, which is displayed as a red asterisk. Active Inference then chooses the action in a fixed order as with the framework of Pezzato et al. [74]. The action with the lowest free energy is always picked in both figures.

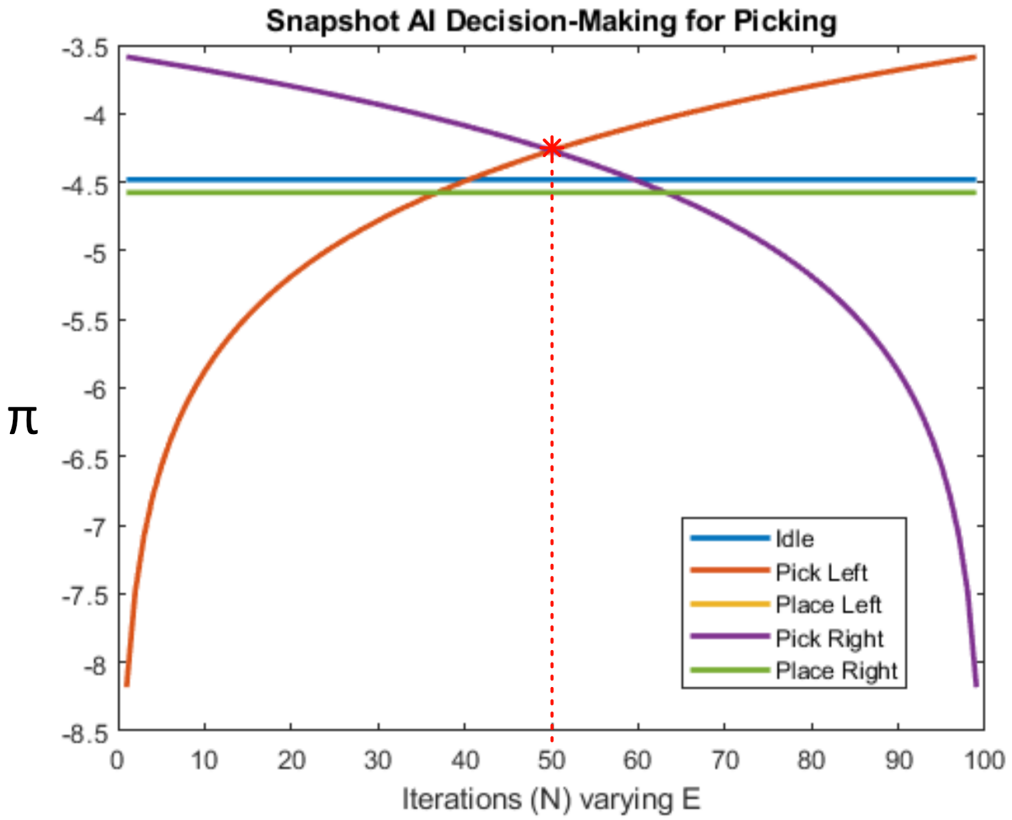


Figure 4-6: Snapshot of decision-making through Active Inference for picking. The prior over plans for picking with the left gripper are modelled to increase during each iteration with a value of 0.01 leading up to maximum likelihood of success (value 1). The right gripper is modelled the opposite, starting at the maximum likelihood of success and decreasing during each iteration until it comes close to zero. The x-axis divided by 100 equals the prior over plans.

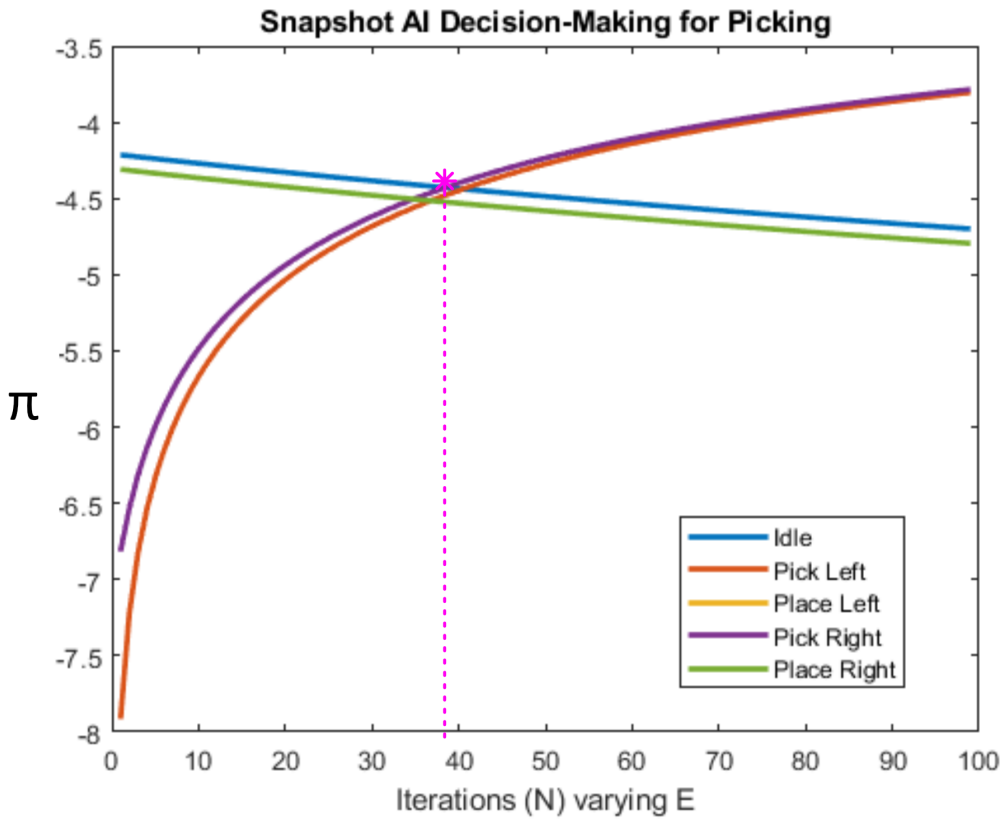


Figure 4-7: Snapshot of decision-making through Active Inference for picking. The prior over plans for picking with the left and right gripper are modelled to increase during each iteration with a value of 0.01 leading up to maximum likelihood of success (value 1). The x-axis divided by 100 equals the prior over plans of each action.

4-4 Airet Ontology

4-4-1 The Need for Standardised Ontologies

Early works of this century on probabilistic reasoning focused on the problem of too many ontologies being available, lacking standardisation, and no mapping existed between similar ontologies. This made comparing ontologies in terms of their coverage of knowledge and reasoning capabilities difficult. Furthermore, strength of ontologies lie in their reuse capabilities, of which reuse focused mostly on applications rather than domains. This allowed for many ontologies to exist with overlapping definitions within the same domain, making selection of concepts for reuse a task prone to errors.

Ontologies have gained popularity in the AI community as a way to create explicit formal semantic shareable knowledge. It comes with the freedom of naming and classifications provided by creating ontologies. A solution to the problems mentioned above was thought to be the creation of probabilistic mappings between ontologies, with a measure of similarity between them and the end-purpose being enabling semantic reasoning. Several of these methods that provide a probabilistic mapping are GLUE [27], CAIMAN [54], an ontology mapper for BayesOWL [68] and OMEN [61]. More on different types of mappers can be found in the survey [64]. Ontology mappers often still required expert knowledge and were prone to mistakes. Another approach that complements mapping between ontologies during this time was creating standardised foundational ontologies with formal definitions that are reusable and extendable for domain-specific ontologies. This approach minimised the errors for ontology mapping by defining higher-level concepts and properties so that these fundamental concepts are understood by their respective scientific communities. Examples of these concepts are Object, Task, Action, Agent, and Component. Examples of properties are member, robotPart, dependsOn and interactsWith.

4-4-2 Ontologies for Airet

The Airet ontology is based on the PMK ontology, which is in turn based on the upper-level ontologies SUMO, CORA, CORAX and ROA. Suggested Upper Merged Ontology (SUMO), [63] is an upper-level ontology created by *the standard upper ontology working group* with collaborates in several fields of engineering, philosophy and information science, sponsored by IEEE. Cora was proposed to standardise core definition in robotics and automation, is sponsored by the IEEE, and depends on concepts defined in SUMO [78]. Cora was extended to include CORAX [29], which added concepts about the physical environment and agent-agent interaction. Furthermore, the object property dependsOn was added, which allows for denoting a dependency, which could, for example, be the dependency of an object on a region or environment. RPARTS provides general concepts on robot parts, including the purpose of parts, for example, parts for sensing, Acting and communicating. POS provides the main concepts and relations underlying the notions of position, orientation and pose. These mentioned extensions can be found in [29]. More recently, CORA has been enhanced by adding the knowledge called Autonomous Robot Architecture Ontology (ROA) [67], which defines the main concepts and relations regarding robot architecture for autonomous robots systems. ROA provides the definitions of behaviour, function, goal, and task concepts [67]. The terminology used in the Airet ontology can be found in the table 4-1. In table 4-2 one

finds the coverage of the Airet ontology for relevant terms in the autonomous robotics domain. Prior works have used this list of terms as a scale consisting of agreed terms in this domain needed for autonomous ontological reasoning [65, 23]. As for the reasoning scope, these works in the autonomous robotics domain use the capabilities needed for cognition as defined by langley et al. as a measure of cognition provided through reasoning on the ontology [55, 90]. The cognitive capabilities covered by the Airet framework can be found in figure 4-3

4-4-3 Airet Terminology

Definitions of concepts of the PMK ontology [24] can be found in appendix 7-2

PickVacBox: Action type specifying the action to be performed, the controller (in this case, a vacuum gripper) and the type of product, which in this case is of box-type.
PickServoBox: Action type specifying the action to be performed, the controller (in this case a servo-electric gripper) and the type of product, which in this case is of box-type.
MoveBase: Action moves the base of a robot, leading to translation and/or rotating of the robot's whole body.
ActionType move: Class for grouping action types that deal with translation and rotation of components and/or subsystems.
ActionType pick-place: Manipulation of an object by translating and/or rotating to grasp and hold an object or the transverse of this manipulation, releasing an object with the desired location and/or orientation.
Under class robot add TIAGo++: A robot class for types of robots associated with the TIAGo++ system of Pal-Robotics. The individuals of these classes can have different components like single arms, dual arms, servo-electric grippers and the Hey5 robotic fingers.

Under robot components, next to the preexisting arm and sensors, two-component OWL subclasses have been created:
Gripper: End-effector with the capability upon actuation to manipulate objects.
mobileBase: component class to specify components resulting in translation and rotation of the robot.

Under the class constantQuantity, a subclass AlgorithParameters has been created. AlgorithmParameters: a class to specify parameters that can be considered constants. Example parameters are controller targets like target velocities, accelerations and joint torques. Under the class AlgorithmParameters, a subclass ActiveInferenceParam is created. ActiveInferenceParam: subclass specifying parameters to initialise Active Inference and adapt based on task, (robotic) agent, working environment and prior knowledge available. Two subclasses which are used in this work are EncodedDesires: containing translations from task specification to desired states to be perceived by the agent. PriorOverPlans: Containing the agent's beliefs over available plans that can be selected to satisfy encoded desires. In chapter 2 this is also denoted by the values of E. Friston, the founder of Active Inference, who refers to these values as habits [81].

Under the PMK class GripConstraints, a few constraint classes have been modelled. Some of these are: MaxGripWeight: subclass of maximum weight constraints that gripper components can handle. Individuals of these classes could be payloads of end-effectors, like a servo-electric gripper that can pick objects with a maximum weight. MaxGripWidth: subclass of maximum gripper width constraints that gripper components can handle or the gripping task permits.

Individuals of these classes could be maximum extensions of end-effectors, like a servo-electric gripper being able to pick objects with a maximum width only due to actuators not extending beyond this width. Other individual constraints could depend on the application, e.g. the shelf width in a retail store. MinGripWidth: subclass of minimum gripper width constraints that gripper components require for successful gripping task performance. Individuals of these classes could be the minimum width vacuum grippers needed for proper suction of objects for grasping.

Airet Object properties extended The class dependsOn is taken from the CORAX [29] an extension of CORA standardised upper ontology. It specifies the directional dependency of physical entities. An example is the ActionType pick class depending on a physical entity class arm. This definition adheres to standardisation and aligns with the definition created by BayesOWL given in section 3-3-2.

hasActionType hasPriorOverPlans: An object property related to the algorithm Active Inference specifies the relation of having a prior over plans fitting the task at hand. It can specify which actions are more favoured over others. Examples are the individual task "Pick a box product with a servo-electric gripper, and a robot TIAGo++ with individual Tiago1" has prior over plans ErightArmTiago. This prior specifies the chance of the right arm of this specific robot functioning adequately.

Airet Data properties extended AlgorithmParam: A data property specification of parameters needed for specific algorithms. ActiveInferenceParam, a sub-property of AlgorithmParam, is a data property specifying parameters needed by the Active Inference algorithm introduced in chapter 2. ValuePriorPlan is a sub-property of ActiveInferenceParam, specifying a value of the format double, containing the prior over plans E.

4-4-4 Implementation

The Airet ontology is designed using ontology web language (OWL) with Protégé ontology editor [2]. Since Airet is modelled using the web ontology language OWL, which is created with description logics in mind, a description logic reasoner is preferred. The DL logic reasoner Pellet is used for reasoning. Pellet was the first reasoner that supported all of OWL DL [80, 22]. Furthermore, the pellet supports SWRL rules (SWRL includes a high-level abstract syntax for Horn-like rules)[3]. Some drawbacks that description logics (DL) suffer from are that all the knowledge must be represented on the abstract logical level. In many applications, one would like to be able to refer to concrete domains and predefined predicates on these domains when defining concepts. An example of such a concrete domain could be the set of nonnegative integers, with predicates such as $\geq$ (greater-or-equal) or $<$ (less-than). Implementing these predicates in DL is possible. However, they often require defining concepts in the ontology to make reasoning on them possible. Coding languages do not suffer from this limitation. Hence a framework supplying reasoning with DL, with the programming language Python, is a suitable choice. This framework OWLREADY2, facilitator of reasoners pellet and hermit for python, translates OWL concepts into Python classes, extending the reasoning operations applicable. *Upon loading the concepts and properties taken from the PMK ontology into the reasoner*, major problematic definitions in the released ontology were identified. These were

[2]http://protege.stanford.edu
[3]https://www.w3.org/Submission/SWRL/

	Classes	Object Properties	Data Properties
SUMO	• Quantity • Attribute • Measuring Device • Collection • Artifact • Region	• member • located • time	
CORA	• Robot • Robot Group	• robotPart	
CORAX	• Physical Environment	• dependsOn • interactsWith	
ROA	• Sub-task • Task		
PMK	• Quantity Aggregation • Artifact Component • Robot Component • Measuring Device Group • Measuring Device Component • Semantic Environment • Spatial Context • Temporal Context • Situation • Atomic Function	• hasMass • hasMaterial • hasColor • hasGraspingPose • hasConstraints • hasManConstraints • satisfies • hasPart • isPartOf • isRelatedTo • toTheLeftOf • toTheRightOf • inFrontOf • aboveOf • belowOf	• Color • Dimensions • mass • Material
AIRET	• PickVacBox • PickServBox • PickVacCyllinder • PickServCyllinder • MoveBase • PickPlace • TIAGo • Gripper • MobileBase • Algorithm Parameters • ActiveInference Parameters • Encoded Desires • Prior Over Policies • MaxGripWeight • MaxGripWidth • MinGripWidth	• hasActionType • hasPriorOverPlans	• AlgorithmParam • ActiveInferenceDataParam • ValuePriorPlan

Table 4-1: Table containing the classes, object and data properties of the Airet ontology and their original standardised (upper) ontologies.

Concept	Defined
Objects	**Yes**
Environment map	**Yes**
Affordance	No
Action	**Yes**
Task	**Yes**
Activity	No
Behavior	No
Function	**Yes**
Plan	No
Method	No
Capability	**Yes**
Skill	No
Hardware	**Yes**
Software	**Yes**
Interaction	No
Communication	No

Table 4-2: List of relevant terms for the autonomous robotics domain [65], and their coverage in the Airet ontology, based on PMK[24]. Yes and No state when the term is or is not covered by the ontology of the specific framework. Note that when the term is needed and taken from the upper ontology used within the framework, and/or when the knowledge is captured using a similar term, it is considered that the term is covered. If the upper ontology contains the term, but it is not used, we consider that the term is not included.

mostly related to data, object properties and concepts being named identically. It violates the rules of OWL for reasoning since no distinction can be made between properties and concepts for reasoning. An example is a class On, which indicates a relation between artefacts, being defined as an object property and an OWL concept class in the Tbox of the ontology. After several attempts, it was decided to rebuild the Airet ontology and redefine parts taken from the PMK ontology.

4-5 Airet Reasoner

The Airet reasoner creates context-awareness through reasoning on the environment, the robot and its components and capabilities, the task and the action.

In figure 4-8, one can see the working principle of the reasoner. As was discussed in section 4-2-2, the reasoner is called through the reasoner action node implemented in the behaviour tree. The reasoning action nodes have as input the robot type, individual robot name as defined in the ontology, the desired state of Active Inference to be reached (task goal) and parameters necessary for task execution (for example, object individually to be manipulated).

The robot type is used to identify the main components of the robot through ontological reasoning. For example, a robot can have a move base, camera, lidar sensors, and an arm.

The concepts related to the robot individual are the amount and types of arms and end-effectors.

The next part of the reasoner deals with hard constraints related to the action, manipulation and the environment. The reasoning for these hard constraints is formulated in Description Logics, and depicted in figure 4-10 and 4-11. Figure 4-11 provides detail on the reasoning of manipulation constraints & environmental constraints, used in figure 4-10. Examples of hard constraints are the object width and weight not being allowed to exceed gripper width & payload for the servo-electric gripper. For the vacuum gripper, the minimum width is set as a constraint due to the sufficiently large gripping area needed for suction. An example of an environmental constraint is that the object has to fit into the shelf for the task being stocking a shelf. Some action constraints taken into account are the actions belonging to an action type as defined in the ontology, see section 4-4-3.

- An example action type could be PickVacuumGripperCillinderProd. It is an ActionType class for picking cylindrical products with a vacuum gripper. These action types are relevant for constructing the nodes of the Bayesian Network. The Airet reasoner selects the appropriate action type based on the task, object (belonging to a shape group) and robot individual.
- An example of environment constraint that can be taken into account is the shelf width and height. Products which violate these dimensions cannot be stocked on the shelf, and the task should be abandoned.
- Manipulation constraints pertain mostly to the end-effector and the object properties. Examples are the payload of a servo-electric gripper being larger than the weight of an object to be picked.

If any constraint so far is violated, the prior over plans for the actuators satisfying actions leading to desired states are set to approximately zero, whilst the prior over plans of other actions like staying idle is still higher than zero. It means that the free energy minimisation process through Active Inference will choose the plans that lower the Free Energy the most in the future, from what is observed currently, which in the case of robot manipulation will be returning action idle.

Any other actions like varieties of actions completing pick and place will not lead to task succession. It can be interpreted as the situation (the robot, task and the environment) making action success impossible.

If all the constraints are satisfied, a Bayesian Network is generated from the information of the ontology to provide the prior over plans of suitable actions for successful task completion. This execution scheme is depicted in figure 4-8 and through the pseudo-code for a picking action in algorithm 3.

Bayesian Network formation using Airet ontology

The Bayesian Network is generated through the ActionType concept in the OWL ontology, the object property hasPriorOverPlans, data property PriorOverPlans and the relevant actions specified through the behaviour tree. Continuing on the ActionType class PickVacuumGripperCillinderProd; this would be the relevant Actiontype individual for picking a

cylindrical shaped object with the robot TIAGo++ having ID 1, and a vacuum end-effector. It would then encode individual E values relevant for task succession using the data property defined being ValuePriorPlans. Through using an ontology, the Bayesian Network can be constructed automatically. Furthermore, subnodes can be generated using the same object property hasPriorOverPlans. See figure 4-9.

The assumptions governing Bayesian Networks align with the Markov blanket assumption governing the Active Inference algorithm, in chapter 2. The Markov boundary of a node in a Bayesian network is the set of nodes composed of this node, the parents of this node, the children of this node and this node's children's other parents. The assumption governing Bayesian Networks is called Local Markov Independence, which states that each variable in the joint distribution (which corresponds to a node in the graph) is independent of its non-descendants, given its parents in the same graph, if a joint distribution factorises concerning this directed graph. An example detailing this assumption is that factor VacuumGripTiago is independent of RightArmTiago, (VacuumGripTiago $\perp\!\!\!\perp$ RightArmTiago), in figure 4-9.

The pseudo-code for constructing a classical Bayesian Network can be found in algorithm 1. The Airet framework constructs the Bayesian Networks from the Airet ontology through ontological design and the reasoner, hence modifying algorithm 1. This modified construction of Bayesian Networks can be found in algorithm 2. The main difference between algorithm 1 and 2 is that relations between child nodes and parent nodes are directed through the OWL object property hasPriorOverPlans and that instead of conditional probabilities, the prior over plans are obtained. These are a subset of conditional probabilities, adhering to the same mathematical relations and restrictions, added that they belong to parameters for Active Inference.

Algorithm 1 Classical Bayesian Network Construction

1: Select a set of relevant of variables
2: Order these variables as $(x_1, x_2 ... x_N)$
3: **for** i=1 to N **do** ▷ All nodes in BN
4: Add node x_i to the graph
5: Set parents(x_i) to be the minimal subset of $\{x_1, ... x_{i-1}\}$, such that x_i is conditionally independent of all other members of $\{x_1, ... x_{i-1}\}$ given parents(x_i)
6: Define Conditional Probability Tables for $P(x_i|$ assignment of parents(x_i))
7: **end for**

After construction of the Bayesian Network graph, the final prior over plans denoting likelihood of an action succession is obtained through formula 4-1, where pa stands for parent node, E stands for prior over plans and V is the set of factors influencing the action (E_{Action}).

$$E_{Action}(x) = \Pi_{i \in V} E_i(x_i | x_{pa(i)}) \tag{4-1}$$

Algorithm 2 Airet Bayesian Network Construction

1: Choose a set of relevant parent node variables for ActionType from individuals of OWL class PriorOverPlans
2: Connect these chosen set of variables to Actiontype through *Object Property* hasPriorOverPlans
3: Connect the children of parent nodes to parents through *Object Property* hasPriorOverPlans
4: Order these parent variables as $(x_1, x_2 ... x_N)$
5: **for** i=1 to N **do** ▷ Parent nodes
6: Set x_i to be the minimal subset of $\{x_1, ... x_{i-1}\}$, such that x_i is conditionally independent of all other members of $\{x_1, ... x_{i-1}\}$
7: Add parent node x_i to the BN graph constructed by the Airet reasoner
8: **for** j=1 to k **do** ▷ Children nodes per parent
9: Set $x_{i,j}$ to be the minimal subset of $\{x_{1,j}, ... x_{i-1,j}\}$ such that $x_{i,j}$ is conditionally independent of all other members of $\{x_1, ... x_{i-1}\}$ and $\{x_{1,j}, ... x_{i-1,j}\}$
10: Add Child node $x_{i,j}$ to the BN graph constructed by the Airet reasoner
11: Obtain PriorOverPlans through extracting *Data Property* ValuePriorPlan for $P(x_i|$ assignment of parents$(x_i))$
12: Obtain PriorOverPlans through extracting *Data Property* ValuePriorPlan for $P(x_{i,j}|x_i)$
13: **end for**
14: **end for**

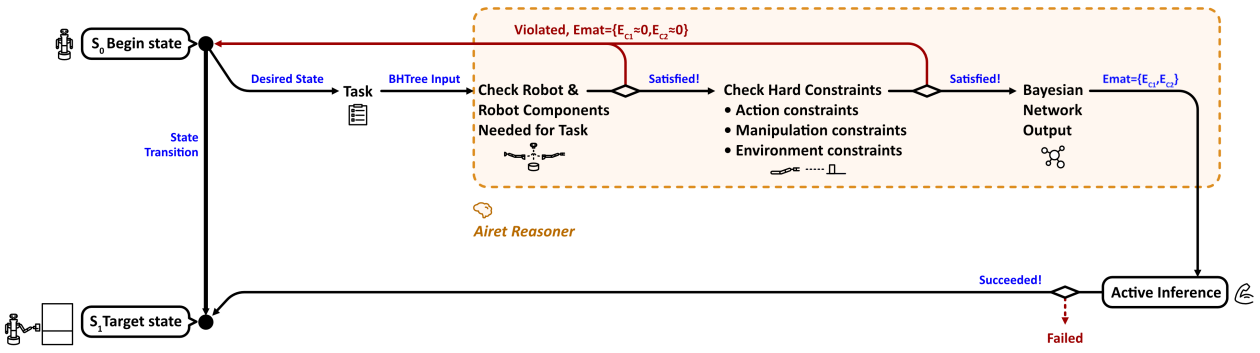


Figure 4-8: The reasoner action loop

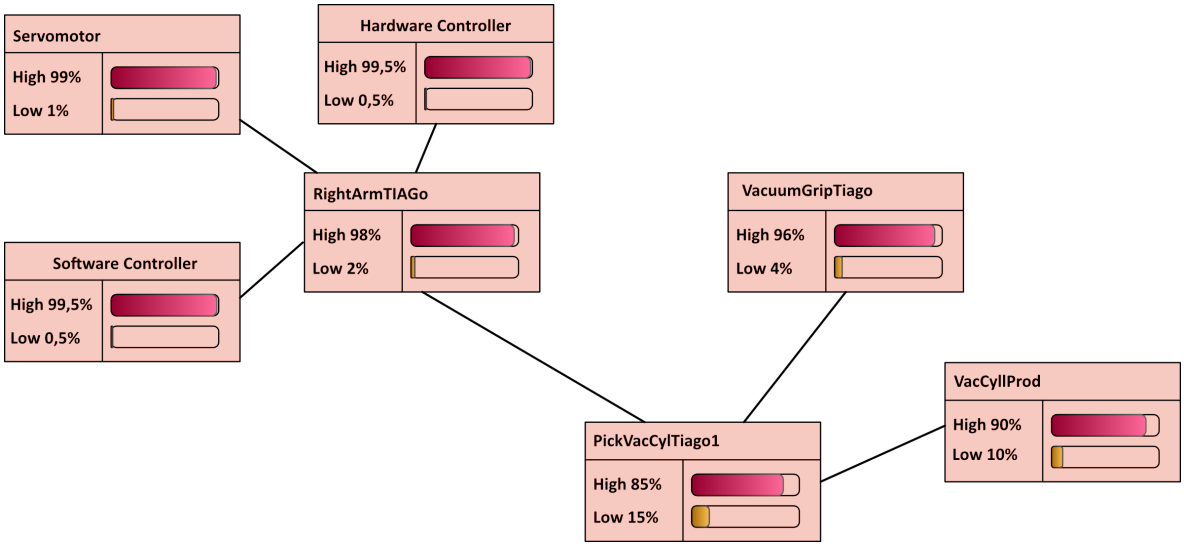


Figure 4-9: Example Bayesian network generated by the reasoner, using concepts for action taking defined in the ontology. The nodes of the Bayesian network contain effects influencing the final action, being a vacuum gripper part of the robot TIAGo++ picking a cylindrical object. Failure modes are taken into account. In this example, the right arm of TIAGo++ functioning properly for task success depends on the servomotor in the arm, the hardware controller and the software controller. The relationship between nodes is created through the object property hasPriorOverPlans. The Bayesian Network is automatically constructed by the Airet reasoner using the Airet ontology.

	Planning & Problem Solving	**Execution and Action**	**Decision Making & Choice**	Learning, Recalling and Reflection	Reasoning and Belief Updating	Perception & Situational Analysis	Predicting & monitoring	Recognition & Classification	Interaction and Communication	Fault Detection, Recovery & Adaptation	Maintained	Use Case
Airet Framework	✓	✓	✓	✓	✓	✓	-	-	-	✓	Yes	Robotic Manipulators (Service Robots for Retail)

Table 4-3: Cognitive capabilities satisfied by the Airet framework, system architectures should contain autonomous mental capabilities as discovered by Langley et al. [55, 90]. These criteria have been used in the autonomous robotics domain for ontology-based approaches as the industry standard for assessing cognition [65]. The underlined cognitive capability *Fault Detection, Recovery & Adaptation* is not a standalone capability but rather a combination of the other cognitive capabilities. Nevertheless, it has been added by the author because it is crucial for autonomous behaviour in a dynamic environment, such as retail.

Pick:-

hasActionType(PickGripperProducttype, Action)∧

∃hasSuperClass(ActionType,Pick)

∃hasSuperclass(Artifact, Producttype)

∧∀hasTaskTarget(Pick, Artifact)

∧∃isPartOf(Region, Workspace)

∧∃hasGraspingPose(Artifact, GraspingPose)

∧∃hasPart(Robot,Arm)

∧∃hasPart(Robot, Sensors)

∧∃hasPart(Robot,Battery)

∧∃hasPart(Robot,Gripper)

∧∃hasPartSide(Gripper,Side)

∧∃hasManipulationConstraints(Width,MaxShelfWidth)

∧∃hasManipulationConstraints(Height,MaxShelfHeight)

∧∃hasArtifactManipulationConstraints(Artifact, Properties)

∧∃hasArmManipulationConstraints(Arm, ArmConstraints)

∧∃hasGripperManipulationConstraints(Gripper, GripConstraints)

MoveTo:-

hasActionType(MoveBase, Action)∧

∃hasSuperClass(ActionType, Move)

∧∀hasTaskTarget(Move, Pose)

∧∃isPartOf(Region, Workspace)

∧∃hasPart(Robot, Drive)

∧∃hasPart(Robot, Sensors)

∧∃hasPart(Robot,Battery)

Figure 4-10: Reasoning formulated in Description Logic supported by the Airet ontology & Reasoner on actions Pick and Move

Figure 4-11: Reasoning formulated in Description Logic supported by the Airet ontology & Reasoner on hard action constraints

Task Failure

Figure 4-5 illustrates an extended behaviour tree including Active inference action nodes and reasoning nodes prior to these action nodes. Within the Airet framework, several methods exist to handle faults. The task planner (behaviour tree) handles failure through the implementation of fallback nodes, including actions executed in case of task failure as with standard behaviour trees. The Action Planning framework used by Pezzato et al. [74] furthermore encodes pre- and post-conditions, which are given priority to be satisfied, if not satisfied before, by identifying a mismatch between belief states and observed states and setting priority to satisfy pre- and post-conditions higher than the desired task. It allows for reactivity to unforeseen contingencies in planning. The final method of failure recovery is handled through the Airet reasoner, which interprets failure and gives priority to working controllers by influencing the decision-making of Active inference through the prior over plans of action. In case of task failure, the fallback node in figure 4-5 of the behaviour tree triggers an update of the Airet ontology. It could, for example, be setting $E_{servomotor}$ equal to zero when fault detection and isolation schemes identify malfunctioning of the right arm servomotor. Since all the prior over policies affecting an action are multiplied, see equation 4-1, the resulting prior over plans of ActionType PickVacCylTiago1 would be set equal to zero, meaning the action can not be executed with the right arm. If the robot has two arms and the other arm functions properly, the priority over plans of the other arm would be closer to a value of 1, meaning this action would be preferred hence facilitating adaptation. Since the priors over plans of each factor influencing action selection are stored in the Airet ontology, a more representative likelihood of success is updated for that action primitive (i.e. $E_{servomotor}$ being a subnode of ERightArmTIAGo is lowered to approximately 0). For this to be achieved, one can use fault detection and isolation schemes discussed in section 3-1-1. These methods can specify component failure, to which the likelihood of success for that action primitive is lowered.

Algorithm 3 Simplified pseudo-code reasoner for pickaction. "A" is the set of actions available to the robot, " V" is the set of factors influencing the action and are the parent nodes of E_i

Require: Input ← [*Robot*; *RobotInstance*; *Action*; $TaskParam$ (*e.g. objectID*)]

Try:
$Action \in A$
$E_{Action} = 1$ ▷ Best case Action possible
for All Robot components needed for Pick Action **do**
  if check Robot Individual has components **then**
    if check if Robot Individual Components have no error upon startup robot **then**
      Get(EnvironmentConstraints)
      Get(ArtifactManipulationConstraints)
      if $ShelfDimensions \in EnvironmentConstraints > ObjectDimensions \in ArtifactManipulationConstraints$ **then**
        Get(GripperTypes) & Get(gripperType, Amount)
        for $GripperTypes, ArmTypes \in sides$ **do**
          Get(GripperManipulationConstraints(GripperTypes,RobotIndividual))
          Get(ArmManipulationConstraints(ArmType,RobotIndividual))
          if $GripperType \in GripperTypes$ & $GripperSpecificAmount > 0$ **then**
            if $ObjectProperties \in ArtifactManipulationConstraints < GripperManipulationConstraints$ **then**
              Get(GripperActionType(Action, GripperType, RobotIndividual))

              for GripperActionType() **do**
                $Get(E_i)$
                $E_{Action}(x) = \Pi_{i \in V} E_i(x_i | x_{pa(i)})$
              end for
            else $E_{Action} = 0$ ▷ Worst case, action infeasible
            end if
          else $E_{Action} = 0$
          end if
        end for
      else $E_{Action} = 0$
      end if
    else $E_{Action} = 0$
    end if
  else $E_{Action} = 0$
  end if
end for

Except: reasoner error {ERRORTYPE} occured. Missing components for action, or action not supported by reasoner

4-5-1 Airet Reasoner Example

In the following simplified example, the decision-making through Active Inference & the Airet framework is demonstrated when considering a dynamically created prior over plans. Assume an Active inference agent that operates in a simplified environment. This agent has two plans, π_1 & π_2, the first being picking up an object with a servo-electric gripper and the second picking up with a vacuum gripper. The agent prefers observation encoded in matrix C. Furthermore, assume the object to be picked is a cereal box with a width of 20cm. Finally, the servo-electric gripper has a width of a maximum of 15cm, and the vacuum gripper has a minimum width of 4cm. Furthermore, assume the following:

$$A = \begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix},\ C = \begin{bmatrix} 1 \\ 0 \end{bmatrix},\ s_{\tau\pi 1} = \begin{bmatrix} 0.95 \\ 0.05 \end{bmatrix},\ s_{\tau\pi 2} = \begin{bmatrix} 0.05 \\ 0.95 \end{bmatrix} \tag{4-2}$$

First, one checks if the hard constraints are satisfied. In this simplified example, it will be assumed that the object width and the gripper width constraints exist. After the Airet framework establishes that all components needed for ActionType Pick and action PickServoTIAGo1 are present, the width constraints are evaluated by comparing the ArtifactManipulationConstraints to GripperManipulationConstraints. See figure 4-11. For action PickServoTIAGo1, through logic reasoning, it is found that the servo-electric gripper maximum width is smaller than the object width, $MaxServElectricWidth < ObjectWidth \rightarrow E_{PickServoTIAGo1} = 0$ As for action PickVacTIAGo1, through logic reasoning, it is found that the object width is larger than the vacuum gripper minimum width, $MinServElectricWidth < ObjectWidth \rightarrow$ construct Bayesian Network for $E_{PickVacCylTiago1}$.

$$\begin{aligned} E_{PickVacCylTiago1}(x) &= \Pi_{i \in V} E_i(x_i | x_{pa(i)}) \\ &= E_{RightArmTIAGo} * E_{VacuumGripTiago} * E_{VacCylProd} \\ &= E_{Servomotor} * E_{HardwareController} * E_{SoftwareController} * E_{VacuumGripTiago} * E_{VacCylProd} \\ &= 0.99 * 0.995 * 0.995 * 0.96 * 0.90 \\ &= 0.85 \end{aligned}$$

The observations under the two plans are:

$$\boldsymbol{o}_{\tau\pi 1} = A s_{\tau\pi 1} = \begin{bmatrix} 0.86 \\ 0.14 \end{bmatrix},\ \boldsymbol{o}_{\tau\pi 2} = A s_{\tau\pi 2} = \begin{bmatrix} 0.14 \\ 0.86 \end{bmatrix} \tag{4-3}$$

When computing the expected free energy for the plans, one gets:

$$G(\pi_1, \tau) = \boldsymbol{A} s_{\pi_1 \tau}^T [\ln \boldsymbol{A} s_{\pi_1 \tau} - \ln \boldsymbol{C}] - diag(\boldsymbol{A}^T \ln \boldsymbol{A})^T s_{\pi_1 \tau} \tag{4-4}$$

$$\begin{bmatrix} 0.86 \\ 0.14 \end{bmatrix}^T [\ln \begin{bmatrix} 0.86 \\ 0.14 \end{bmatrix} - \ln \begin{bmatrix} 0 \\ 0 \end{bmatrix}] - diag \left(\begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}^T \ln(\begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}) \right)^T \begin{bmatrix} 0.95 \\ 0.05 \end{bmatrix} \approx 36.7615 \tag{4-5}$$

which is the same for each plan. Note that for computation, as is done before in literature [81], a value of 10^{-16} is taken to approximate zero, enabling computation since the logarithm of zero is minus infinity.

Assuming a value for the Variational Free energy $[F(\pi_1), F(\pi_2)]^T$ of $\begin{bmatrix}1.83\\1.83\end{bmatrix}$, This prior over plans as computed before is $E_p = \begin{bmatrix}E_{PickServoTIAGo1}\\E_{PickVacCylTiago1}\end{bmatrix} = \begin{bmatrix}0\\0.85\end{bmatrix}$

$$\boldsymbol{\pi} = \sigma(\ln E_p - F - \gamma G) = \sigma\left(\begin{bmatrix}-36.8414\\-0.1625\end{bmatrix} - \begin{bmatrix}1.83\\1.83\end{bmatrix} - \begin{bmatrix}36.7615\\36.7615\end{bmatrix}\right) \approx \begin{bmatrix}0.01\\0.99\end{bmatrix} \tag{4-6}$$

One can see that the likely plan chosen is the second plan, picking up with a vacuum-gripper (PickVacCylTiago1).

4-6 Summary

This chapter introduces the essential parts of the Airet framework. The Airet framework excels over other ontological reasoning approaches in that it allows for generating context-awareness, including preferences over actions and handling of contingencies. This through a novel approach using Active Inference for Action Planning, behaviour tree for task planning, ontological reasoning through Description logic & Bayesian network for computationally efficient creation of context-awareness, extended to deal with contingencies and action preferences.

The process starts with tasks encoded in the behaviour tree as Active inference desires. Furthermore, the behaviour tree contains reasoning nodes which call upon the Airet reasoner. Besides encoding desired tasks and reasoning nodes, the behaviour tree allows for reactive action selection in case of missing preconditions to actions. The Airet ontology provides terms needed for creating context awareness and adapting in case of contingencies for mobile manipulation in retail. It includes environmental & robot constraints, robot components and capabilities and factors influencing action selection. The Airet reasoner utilises these terms to generate context-awareness, leading to the best feasible actions selected for a task or task rejection if actions are infeasible. The Airet reasoner directly influences the action selection through Active Inference by providing a solid preference for certain actions over others, in the form of a prior over plans. This is through a combination of hard constraints & a Bayesian Network. The hard constraints decide action feasibility upon nominal functioning of components, taking into account the robot, the action and the environment. The Bayesian Network considers factors influencing action success, like contingencies such as component failure & handling of objects by robot components. *The Bayesian network allows for more efficient reasoning & a higher level of modularity & reuse when compared to using a logic-based approach solely. Furthermore, it allows for representing uncertainty in actions which is difficult to implement through first-order logic alone.* This Bayesian network outputs a prior over plans for each action available.

Initially, the Airet framework architecture is introduced, followed by the integration of ontological reasoning into task planning. Next, concepts from important standardised ontologies

and an ontology for mobile manipulation are reused and extended for Action Planning using Active Inference in combination with a Bayesian network, enhancing the decision-making. The Bayesian network is constructed automatically by the Airet reasoner, using the Airet ontology. The ontology design is focused on reuse and the modularity of the code implementation, allowing for significant design & development freedom. The logic base of the Airet reasoner consists of a combination of description logic and the programming language Python. Besides standardisation, a large number of terms for the autonomous robotics domain are covered, together with the most crucial cognitive capabilities for deliberation.

Chapter 5

Results & Case Studies

This section introduces four case studies with six scenarios that demonstrate the benefits of using the novel proposed framework in chapter 4. The first case study shows how, through context-awareness, task rejection occurs when dealing with an ill-posed task, and task succession happens when the robot has a new configuration with end-effectors satisfying the hard constraints. The second case study demonstrates how, in certain scenarios, specific actions are more favourable than others. The Airet reasoner enhances the likelihood of task succession by reasoning on the feasibility of actions and selecting the best action leading to task succession. The third case study demonstrates that under the normal functioning of both controllers and action feasibility, some actions can still be preferred over others. For example, actions with more reliable components are desired over those with less, based on context-reasoning facilitated by the Airet framework. The fourth case study deals with component failure triggering reconfiguration through context awareness provided by the Airet framework. These cases are compared to their counterparts, executing the same scenarios with the task planning framework of Pezzato et al. [74]. This section starts with the system overview for implementing Airet in a retail environment, explaining the function of all the modules supporting and/or facilitating the Airet framework. Next, it explains how the Airet framework influences decision-making with Active Inference, highlighting when an action is chosen over others. Lastly, it presents the case studies are presented and discusses the results in an experimental setup in a real-world retail environment. One case is an exception, which is simulated in a virtual environment due to the lab AI for Retail (AIR) Lab lacking the necessary physical components.

5-1 System Overview

5-1-1 Robot Simulation

The capabilities of the Airet framework are demonstrated through experimental case studies in a real world retail environment, as well as through virtual simulations in a simulated retail environment, see figure 5-1. The robot used in both simulation and real world is the

TIAGo++ Steel version (referred to as TIAGo++), which is the extended version of TIAGo Steel (referred to TIAGo). TIAGo contains a single robotic arm with a servo-electric end effector. TIAGo++, compared to TIAGo, contains dual robotic arms and servo-electric end-effectors instead of a single arm and end-effector. Furthermore, it contains a move-base, a differential drive, a longitudinally translating torso and a pan-tilt head with an RGB-D camera. The robotic arm has 7 degrees of freedom. The base sensors contain lasers and three ultrasounds sensors.

Modelled Vacuum Gripper

At the moment, Pal-robotics, the creator of the TIAGo++ robot, only provides a limited amount of end-effectors. These are the Hey5 fingers, the servo-electric grippers and the Robotiq 2F 85/140. Possibility for extension is available. In a retail environment, one can often encounter objects that contain many sizes and shapes. Objects that are less than 1kg yet bigger in size can be handled by the Hey5 hands and servo-electric grippers, for example, tea boxes. Intuitively, these objects can be better handled with a vacuum gripper. Hence this gripper is modelled in Gazebo. During the research, the TIAGo++ robot available at the AI for Retail (AIR) Lab had component failures. One problem was the overheating of the left-side servo-electric gripper, causing it to shut down. Other failures included arm malfunctions. These failures were only encountered on the left side components.

5-1-2 Software

Discrete Active Inference (AI) package

The Active Inference perception module interprets percepts of the environment in an understandable manner for the Active Inference algorithm. See figure 5-3. This understandable manner includes whether desired states or preconditions have been satisfied. By minimising the Variational Free Energy, it ensures that the belief state of the agent is aligned with observations of the environment. See section 2. The AI templates encode the meaning of these percepts and how actions influence the belief state of the agent, also known as the Markov decision process model parameters, or the prior to the generative models. The AI Action selection module contains the Active Inference loop selecting the appropriate actions to decrease the uncertainty of the agent's environment and satisfy the agent's desired outcome of observations by minimising the Expected Free Energy, see section 2. These desired outcomes of observations, which can be understood as the task plans, are encoded in the behaviour tree. A C++ behaviour tree package is used [1]. These tasks could, for example, be being at a desired location in the supermarket environment or picking a specific product. For communication between packages, ROS service and action nodes are used. See figure 5-3.

[1] https://www.behaviortree.dev/

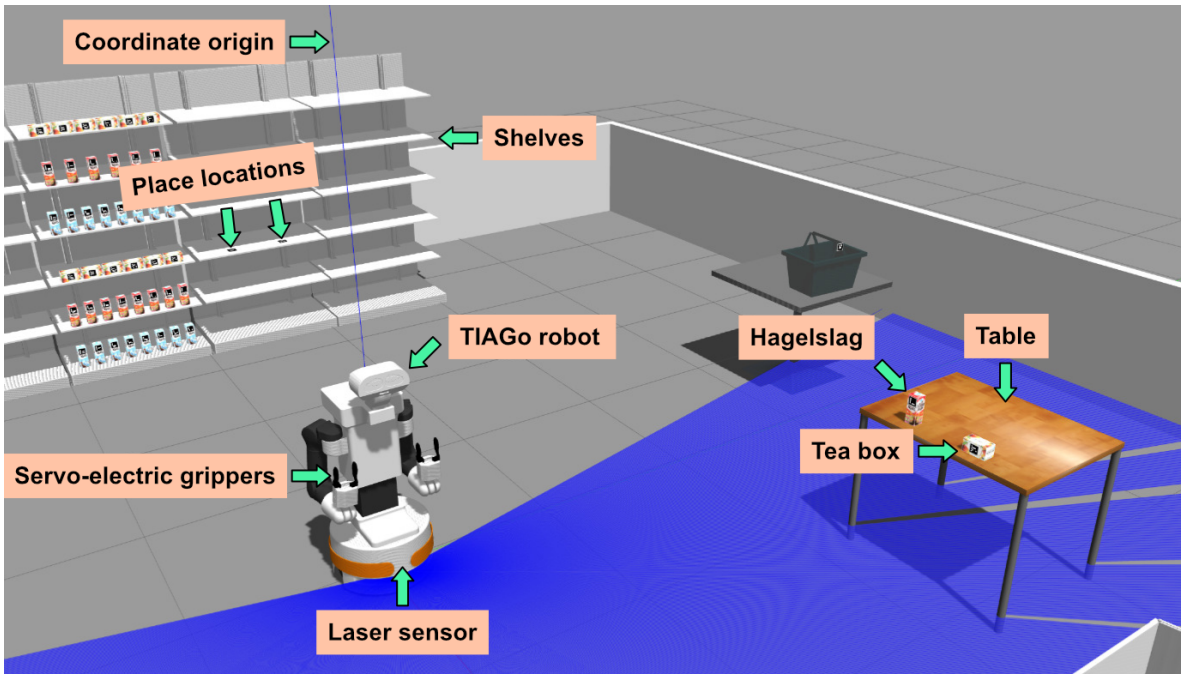


Figure 5-1: Figure displaying the lab environment where the robot TIAGo++ is operating. It displays both stocked and empty shelves, with two ArUco markers marking the place locations of the objects. TIAGo++ in this depiction has dual servo-electric grippers, and lasers are displayed in blue. The table has two products with different properties on top. These products are Hagelslag (Dutch chocolate sprinkles) and a tea box.

Airet Reasoner & Ontology Package

The Airet ontology is constructed using the OWL2 web ontology language [2]. For designing the ontology, Protégé is used [3] which provides a GUI for creating and altering ontology classes & properties. The Airet reasoner is programmed as a service node, which utilises OWLREADY2, a Python ontology-oriented programming package. It can load OWL 2.0 ontologies as Python objects, modify, save and perform reasoning on them [4]. The components forming the reasoner are the 1. CRUD: deals with creating, reading, updating and deleting values in the ontology. It is relevant when modifying object properties, the object locations and priors over plans for action successes. For example, when a component fails, have the ontology reflect the real world by lowering the prior over plans (a factor influencing action-selection) of this component for action-taking. The reasoner server facilitates the service response to the reasoner client node in the Discrete AI package. The reasoner module contains all the reasoning logic and the code to automatically generate Bayesian Networks for decision-making, outputting the prior over plans influencing action selection.

Retail Store Skills Package

The Retail store skills package communicates with the controller clients, sending and receiving feedback on the action performed. The Pick & Place server receives the task, the object

[2]https://www.w3.org/TR/owl2-overview/
[3]https://protege.stanford.edu/
[4]https://owlready2.readthedocs.io/en/v0.37/

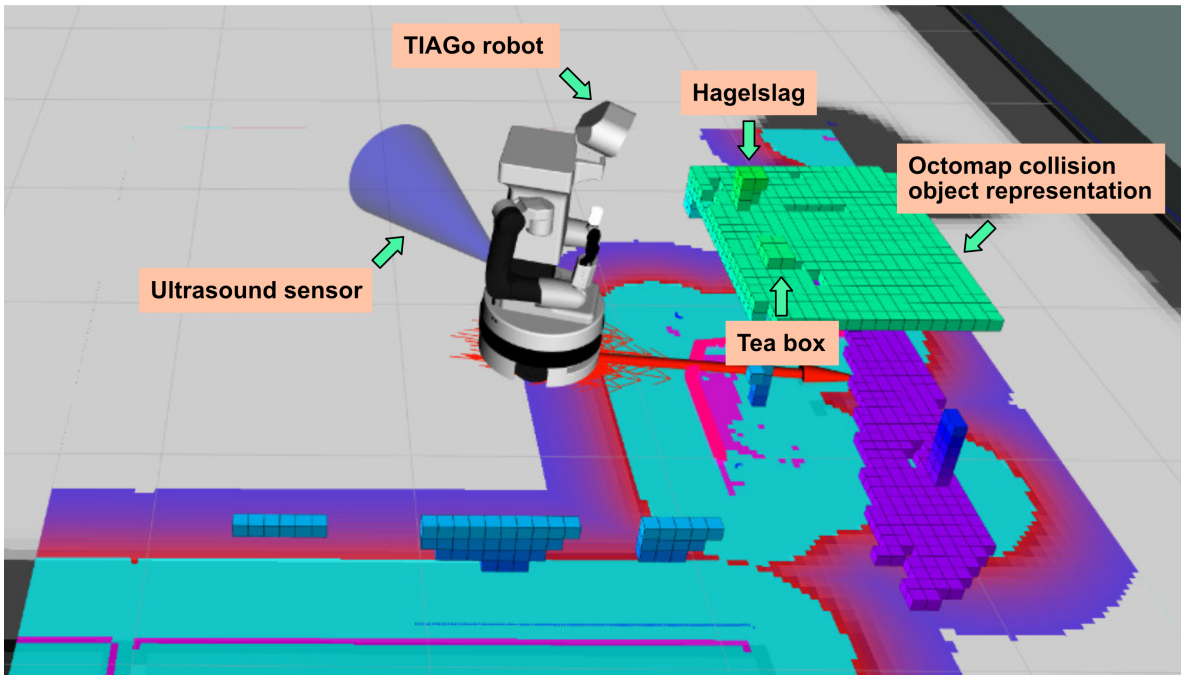


Figure 5-2: Figure displaying the OctoMap in the lab environment where the robot TIAGo++ is operating. It depicts the output of an ultrasound distance sensor as a blue cone. Furthermore, in the lower levels, the distance from obstacles is depicted in purple, red and blue, ranging in this order from less close to obstacles to possibly hitting obstacles. On closer look, one can see the internal model of obstacles the robot has generated through the OctoMap framework and its sensors. The tea box and Hagelslag (dutch sprinkled chocolate) are depicted as blocks, with an uncertainty of their size.

with its location, orientation and ArUco marker ID. Furthermore, it allows for checking the reachability of objects based on object detection and the formation of a motion plan. It is relevant for the perception module of Active Inference since it is a precondition to actions pick and place that needs to be satisfied before these actions can be executed. Furthermore, it communicates with the Grocery store Utils module servers, containing the clients to these servers.

Grocery Store Utils Package

This package is comprised of three ROS Services. The collision obstacle server provides services to manage the creation and deletion of collision obstacles in the Moveit Planningscene. The server uses prior knowledge of the object's geometry and placement of the ArUco marker. The Grasp pose server returns a predefined grasp pose of a certain object type. The grasp pose is specified in the frame of the detected ArUco marker.

Retail Store Simulation Package The Retail store simulation package contains the retail store map where the robot TIAGo++ is operating. This map is generated by allowing the robot to manoeuvre through the lab environment and detect fixed obstacles blocking movements, e.g. shelves, tables and walls. The worlds module contains information needed by ROS Gazebo to generate the world environment. This information includes details on the placement of models (coordinates and rotations), their interaction with each other (e.g. a basket placed on top of a table), world lighting and simulation parameters (e.g. friction coefficients,

gravity coefficient and step-size used by ode integrator). The model's folder contains sdf/dae models of objects in the Gazebo world, including physical parameters like inertia and collision behaviour. When executing this framework on the real robot, this package is replaced by the actual retail store in the real world. The launch files startup the Gazebo world with all the servers this framework requires, including the robot controller servers.

Custom End-effector package This module contains three packages needed for defining and utilising a custom end-effector. The Dual-arm description package contains a simple wrapper to include the URDF description of the custom end-effector, so the robotic system knows its new configuration. The Moveit Configuration package contains configuration files needed for Moveit to interact properly with the new end-effector. The Dual end-effector configuration package contains configuration and launch files used by the TIAGo++ internal architecture. It can be used to generate predefined motions for the TIAGo++ robot. Furthermore, it allows for defining collision constraints for the retail store Gazebo simulation between robotic components (e.g. a robotic end-effector is not allowed to go through the robot's body, as is with the real-world situations).

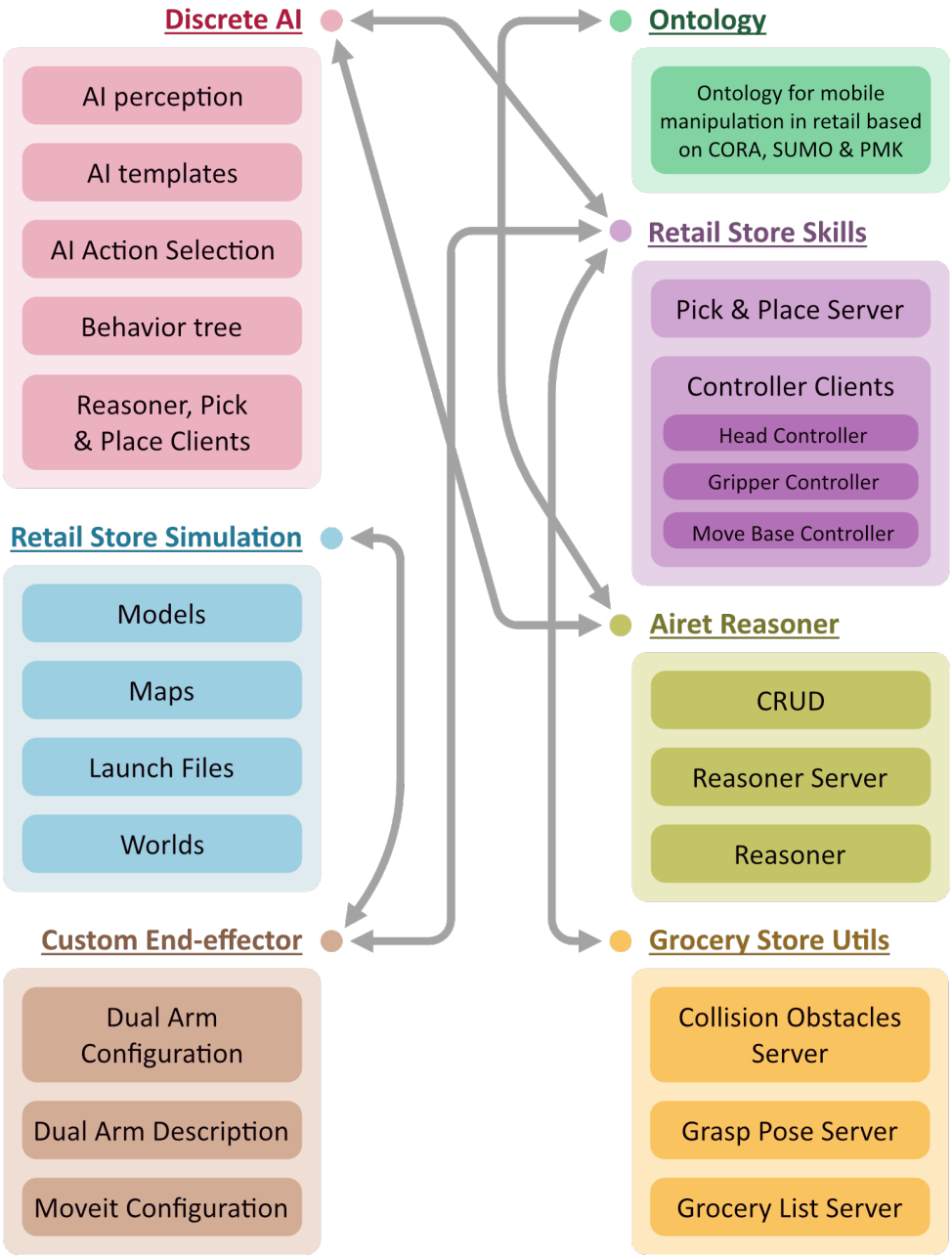


Figure 5-3: This figure provides the System Overview of the packages, the ones fully developed in green (the ontology in apple green and reasoner in olive green) and the other packages modified and extended for context-aware mobile manipulation with Active Inference in a retail environment. Communication between modules of packages is facilitated through the ROS platform.

5-2 Experimental Evaluation

This section demonstrates the capabilities of the Airet framework through four case studies . The first case study contains an ill-posed task, where an infeasible task is set to be satisfied. As for the second case study, it shows how an infeasible task becomes feasible in the presence of end-effectors with inherent differences in capabilities. The third case study shows how certain controllers might be preferred over others despite the feasibility of actions that result in task satisfaction. The final case study demonstrates the adaptation capabilities under controller failure that come from context-awareness through the Airet reasoner. Besides the third case study, the studies are compared to the results from executing the task with the Active Inference for reactive task planning framework of Pezzato el. al [74], as introduced in section 3-1-2. This framework is referred to as baseline.

The input to the Airet reasoner is as follows: [Robot;Robot individual;Task;Task parameters;]. This input is sent from the Behavior Tree reasoning action node, upon tick, in the form of a ROS service call, to the Airet reasoner service with this input. The robot's name is passed to the reasoner to retrieve robot-specific information and constraints. The robot individual is used for robot-specific configurations (e.g. having a single-arm or a dual-arm). The output of the reasoner will be a prior over plans for each task executed with a specific robot instance and controller, which is input into the Active inference algorithm for decision-making.

For example: a robot TIAGo++ with an individual ID defined in the ontology as Tiago1. This robot individual has a left and right arm, with a vacuum gripper and servo-electric gripper, respectively. The task is Pick, and the task parameter is the objectID, a Cereal box (CerealBox1). The input to the Airet reasoner would be $[Tiago; Tiago1; Pick; CerealBox1;]$. The returned priors over plans obtained from the reasoner would be:

$$[E_{Leftarm\ Vacuum\ Gripper}, E_{Rightarm\ Servo-Electric\ Gripper}].$$

5-2-1 Case Study 1: Ill-posed Task

Scenario 1-A: Irrecoverable task failure using the Baseline [74]

As discussed in section 4-3 and chapter 2, Action selection through Active inference will prefer actions that satisfy desired states of the robot since they minimise the uncertainty of the robot with respect to the outer world. However, this view of action selection in the real world is too simplistic. Constraints might be present which render the task infeasible. An example of such a constraint is the artefact to be picked having a width larger than which is feasible to be picked by the robot gripper. Figure 5-4 shows the reasoning-action loop when no reasoner and ontology are present. One can see in this figure that the right arm of the robot containing a servo-electric gripper is chosen by the Active Inference algorithm solely since this is the fixed order in which actions satisfying a desired prior are implemented. The result of lacking the context-awareness facilitated by the ontological reasoning of the Airet framework is irrecoverable task failure. One can see that the cereal box with the exceeding width is tipped in an attempt to pick this product. Especially in human-robot collaborative environments, this behaviour is unwanted as it results in compromised safety, due to the object possibly falling from the shelf. The object could for example shatter if the object packaging is made of glass.

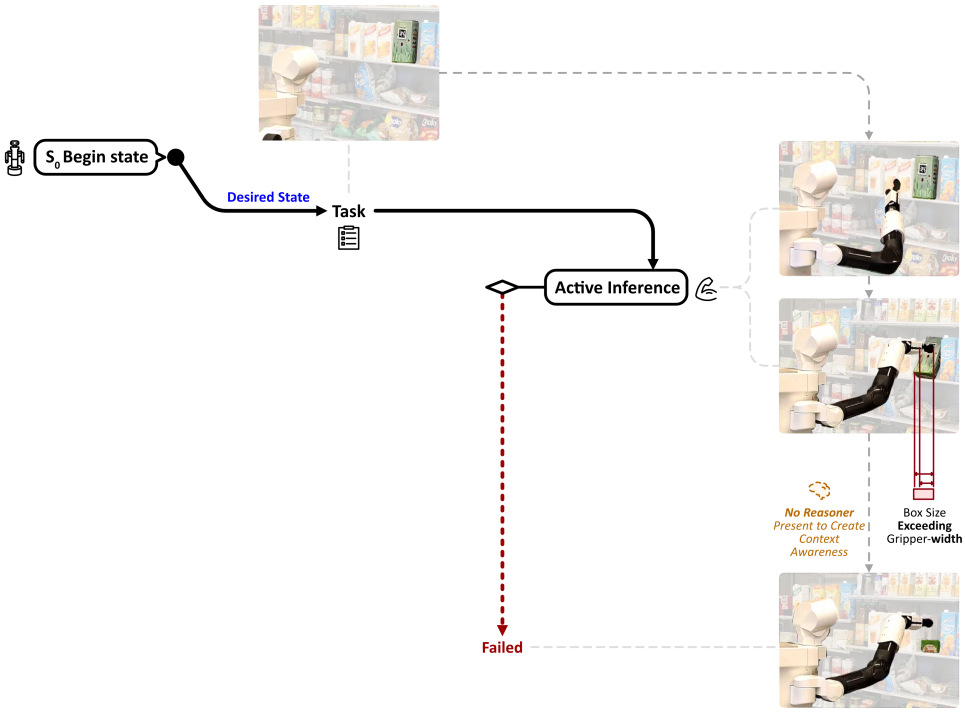


Figure 5-4: Experimental case on the real robot with a picking task, no reasoner being present, and the task being ill-defined. It demonstrates task failure due to a lack of context-awareness, manifested in the tipping of the product.

Scenario 1-B: Task rejection with Airet reasoner & Ontology creating context awareness for decision-making

The same task is given to the system containing the Airet framework. The Airet reasoner correctly identifies the artefact & robot manipulation constraint, being object width exceeding servo-electric gripper width. The only reasonable action with the current setup of TIAGo++, with a servo-electric actuator on both hands, is to stay idle, which is chosen as can be seen in figure 5-5. The result is that the prior over plans for actions for both servo-electric grippers is set to zero. This ensures that the task is recoverable when a robot is present that is equipped with different end-effectors satisfying these constraints or when the human chooses to complete the task.

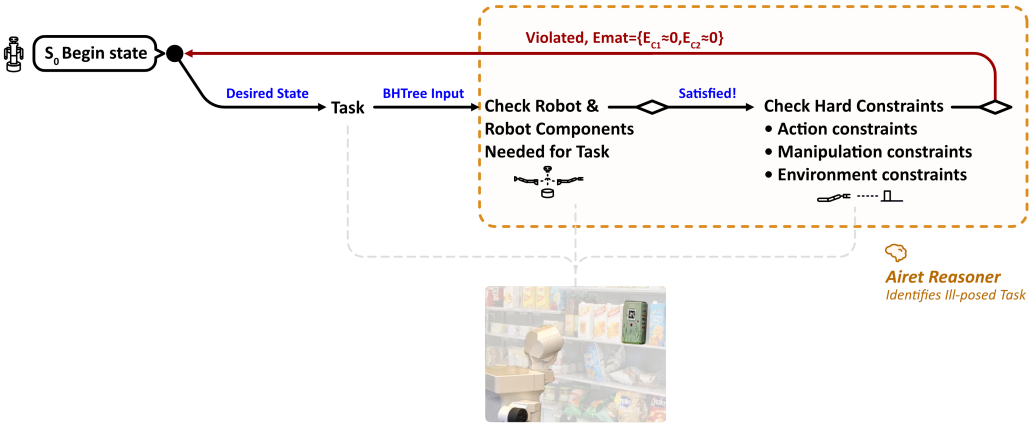


Figure 5-5: Experimental case with a picking task & the task being ill-defined. This case demonstrates the reasoning capabilities of the Airet framework, applied to the real TIAGo++ robot in a retail store environment. The hard constraint pertaining to the object width being smaller than the gripper width of the identified servo-electric gripper is not satisfied. The task is determined to be ill-defined by the Airet reasoner, and the action selection sets the picking with the actions using the servo-electric gripper to be approximately zero. The action of staying idle is chosen instead.

5-2-2 Case Study 2: Choosing Suitable Gripper

Scenario 2-A: Best available gripper for the task, with Airet reasoner & ontology creating context awareness for decision-making

In subsection 5-2-2 the reasoning-action loop of the Airet framework was demonstrated for an ill-defined task. It was found that a servo-electric gripper will not result in successful task completion. When equipping the robot with a vacuum gripper, the Airet reasoner checks the presence and functioning of the robot components needed for the task. The manipulation constraint of maximum gripper width does not apply to the vacuum gripper, although the environmental and action constraints still hold, see figure 5-6. The object must, after all, still fit on the correct shelf, and the robot needs to understand what kind of action it takes. After the hard constraints are satisfied, the Bayesian network is generated for each task executed with a specific robot instance and controller, in this case, picking a box-shaped product with a vacuum gripper. It is done to determine the likelihood of the actuator leading to desired task succession, see figure 5-7. The Bayesian network, next to probabilities related to action succession, also keeps track of the probabilities of proper functioning of the components. During ontology development, using the hasPriorOverPlans object property allows for design freedom for factors the user desires to take into account to influence the decision-making. These factors should, however, be consistent with the factors taken into account for similar actions satisfying the same task. In this case, it was chosen to expand the prior over plans of the left arm of TIAGo++, taking into account the functioning of the components of the left arm. The prior over plans of the action using a vacuum gripper (PickVacuumGripTiago1) resulted in a value of 0.877, while the prior over plans of the action using the servo-electric gripper was set to approximately zero by the reasoner. This value is computed as:

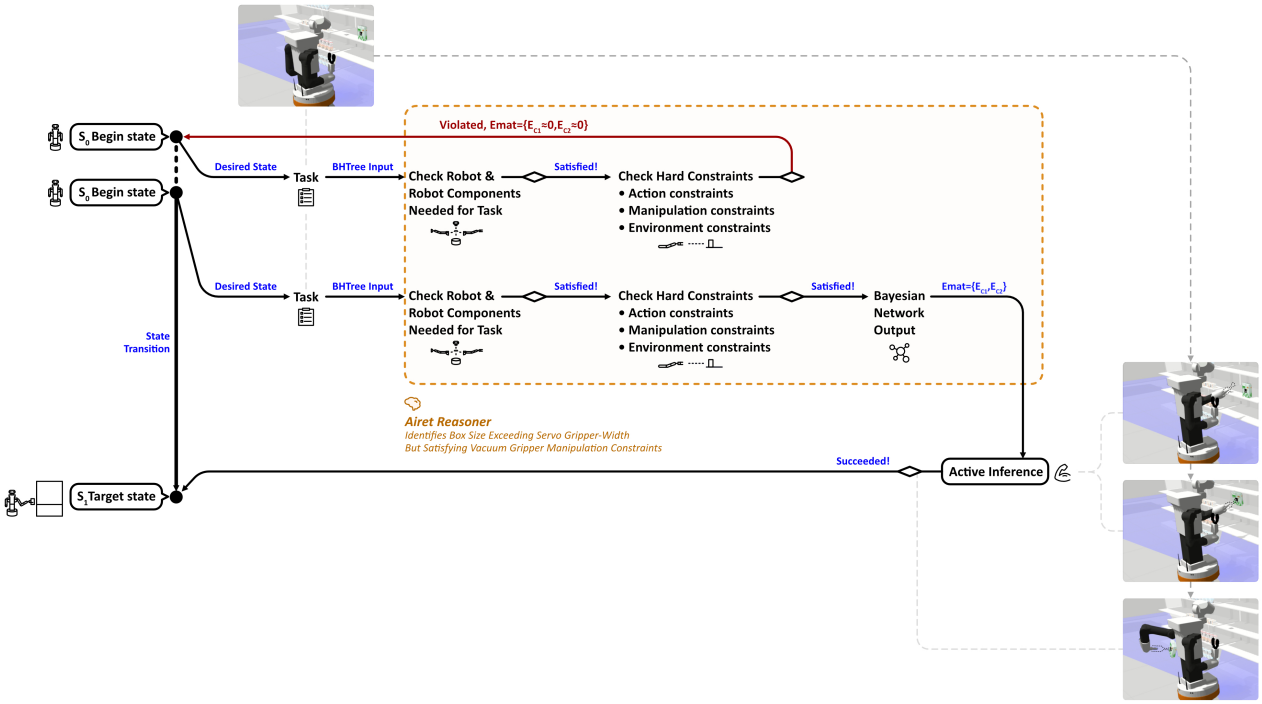


Figure 5-6: Experimental case with a picking task. This case demonstrates the reasoning capabilities of the Airet framework, applied to the real TIAGo++ robot in a retail store environment. The hard constraint pertaining to the object width is not satisfied by the aervo-electric gripper but is satisfied by the Vacuum gripper, together with other constraints like payload. The picking action with the vacuum gripper is executed, resulting in successful task completion.

$$
\begin{aligned}
E_{PickVacGripBoxProdTIAGo1}(x) &= \Pi_{i \in V} E_i(x_i | x_{pa(i)}) \\
&= E_{LeftArmTiago1} * E_{VacuumGripTiago1} * E_{VacBoxProd} \\
&= E_{Servomotor} * E_{HardwareController} * E_{SoftwareController} * E_{RemainderMechanicalParts} \\
&\quad * E_{VacuumGripTiago} * E_{FlexibleMaterial} * E_{SolidSubstance} * E_{UniformMaterialDensity} \\
&\quad * E_{BoxShapedProduct} * E_{VacuumGripAction} \\
&= 0.996 * 0.996 * 0.99 * 0.998 * 0.98 * 0.96 * 0.99 * 0.99 * 0.99 * 0.98 \\
&\approx 0.877
\end{aligned}
$$

In section 4-3 the threshold for choosing an action other than Idle was set to 0.37, which is lower than the prior over plans $E_{PickVacuumGripTiago1}$ of 0.877. Hence, the Airet reasoner deems the vacuum gripper most suitable for the Pick task, resulting in the successful completion of picking the cereal box with the vacuum gripper.

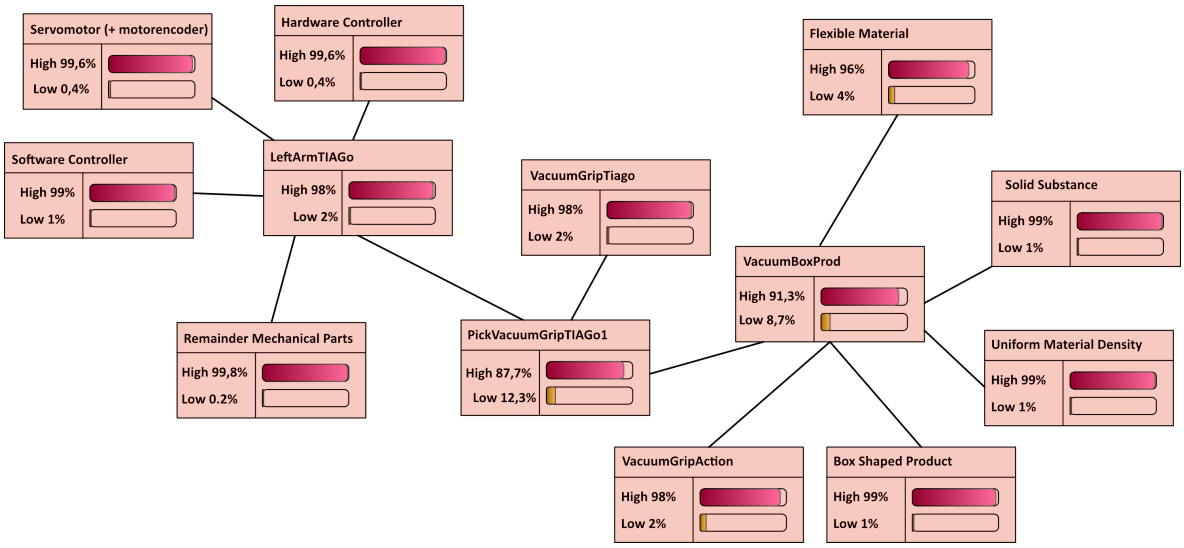


Figure 5-7: Bayesian Network with priors over plans demonstrating the factors influencing the decision making for picking a box-shaped product with a vacuum gripper attached to the left arm. The Bayesian Network is populated through the ontology. Fault detection & isolation methods are assumed to provide information on the failed component.

5-2-3 Case Study 3: Controller Preference

Scenario 3-A: right end-effector preferred

Sometimes, even without failure or unsatisfied constraints, it could be beneficial to select certain controllers over others. For example, one could create a prior over plans tied to the shelf side. It could, for example, result in favouring under normal functioning picking objects located on the left side of a shelf with the left end-effector over the right-side end-effector and vice versa. For example, cylindrical-shaped products might be more desired to be picked with a servo-electric gripper rather than a vacuum gripper since traction can be lost at a round gripping surface. The product shape henceforth also can influence the decision making as is depicted in figure 5-7. Another factor to take into account is the differences in component functioning. It assumes all end-effectors satisfy the hard constraints for picking a specific object; hence both actions could lead to task succession. In practice, it was found that the left servo gripper attached to the TIAGo++ robot in the experimental setup was prone to more frequent component failures during real-world experiments. It could have been caused by wear and tear or inherent factory defects. One could account for this by lowering the prior over plans of the servoGripTiago attached on the left arm through experimental trials or expert knowledge. When this is done, the Bayesian Network of the servo-electric gripper has the original higher value for prior over plans, resulting in this action being more favoured, as is found in the experiment in figure 5-8.

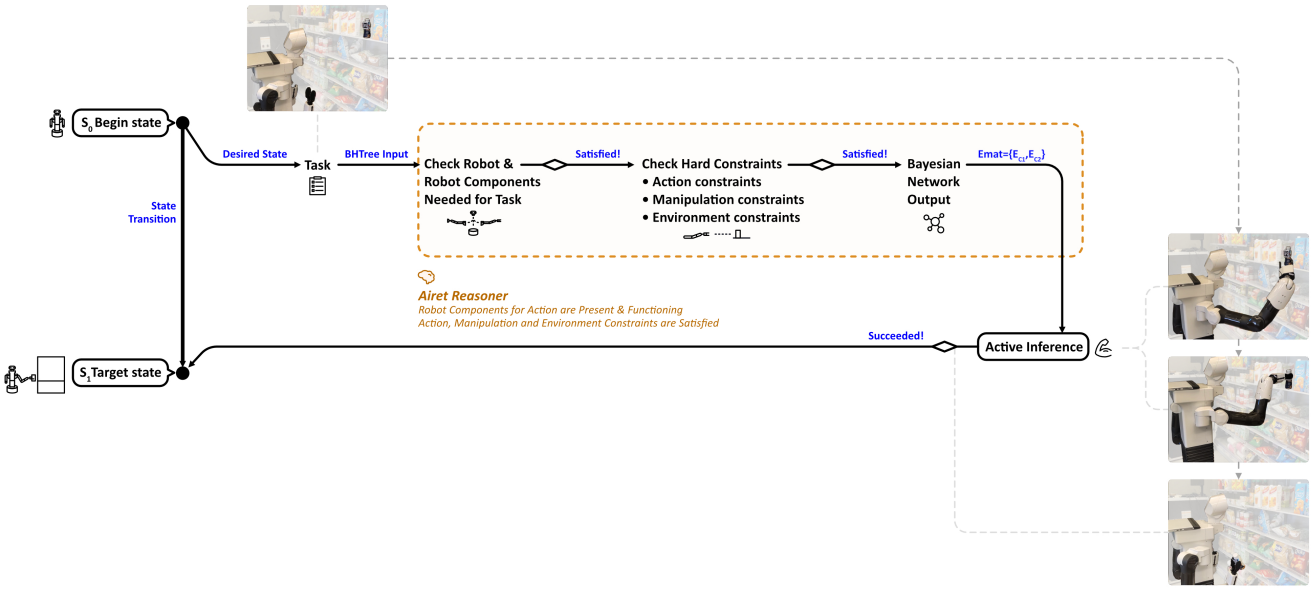


Figure 5-8: Picking experiment demonstrating the reasoning capabilities of the Airet framework, applied on the real TIAGo++ robot in a retail store environment. The right arm and end-effector are preferred over the left ones, despite satisfying the hard constraints and having the right components for task succession. It is due to a larger value of ServoGripTiago for the right arm compared to the left arm. It is due to the left end-effector being more prone to failure due to factory faults and wear & tear. The factors influencing decision-making for the servo-electric gripper of TIAGo++ can be seen in the Bayesian network of figure 5-10.

5-2-4 Case Study 4: Component Failure

Scenario 4-A: Repeated component failure using the reactive Action Planning framework of Pezzato et al. [74]

In a real dynamic world, contingencies can lead to task failure. Task failure due to component failure is a crucial problem with significant consequences for retail. It could be that essential products cannot be stocked, leading to consumer dissatisfaction and loss of profit. When the Airet ontological reasoning is not present to create context awareness, the action selection through Active Inference will repeatedly choose the components for actuation in a fixed order, as long as they are both under normal functioning can result in task succession. It means that the task repeats with the broken controller, resulting in repeated failure. Furthermore, repeated attempts under component failure could lead to irrecoverable task failure and damage to the environment and the robot controllers. It is visualised in a real-world experiment depicted in figure 5-9. One can see that the robot arm fails unbeknown to the Active Inference algorithm for decision-making, to which the same action is attempted repeatedly with repeated failure.

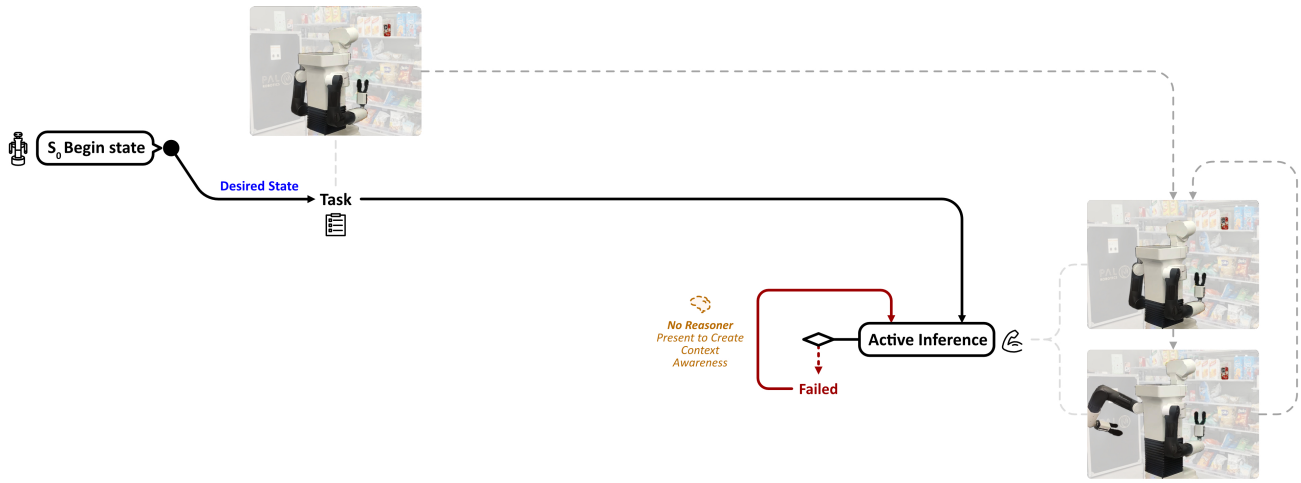


Figure 5-9: Experimental case on the real robot demonstrating the repeated picking task failure when dealing with component failure. This is because the decision-making using Active Inference with the task planning framework [74] selects actions that minimise uncertainty in a fixed order. Context-awareness could distinguish between these actions to prefer functioning controllers, based on reasoning on failure.

Scenario 4-B: Task succession despite component failure with Airet reasoner & ontology creating context awareness

Low-level control signals can be utilised to detect & isolate controller failure. Some methods providing fault detection & isolation through Active Inference were introduced in section 3-1-1. This information can be used to populate the ontology further, hence taking it into account for decision-making with Active Inference. Figure 5-10 shows how the prior over plans constructed by the Airet reasoner for picking a box-shaped product with a servo-electric

gripper attached to the left arm is constructed. It happens after all the hard constraints are checked and evaluated as satisfied. Figure 5-10 also shows that the failure of the servomotor of the left arm renders the whole picking action less likely to succeed. The prior over plans is obtained by multiplying the prior over plans influencing the functioning of the robot's left arm (LeftArmTIAGo) with those influencing the end-effector ServoGripTIAGo with factors related to gripping the object with the end-effector. The resulting value for the prior over plans for this action is 0.0259, which is significantly less than the threshold of 0.37 in section 4-3.

$$
\begin{aligned}
E_{PickServoGripBoxProdTIAGo1}(x) &= \Pi_{i \in V} E_i(x_i | x_{pa(i)}) \\
&= E_{LeftArmTiago1} * E_{ServoGripTiago1} * E_{ServoBoxProd} \\
= E_{Servomotor} * E_{HardwareController} * E_{SoftwareController} * E_{RemainderMechanicalParts} & \\
*E_{ServoGripTiago} * E_{FlexibleMaterial} * E_{LiquidSubstance} * E_{UniformMaterialDensity} & \\
*E_{BoxShapedProduct} * E_{ServoGripAction} & \\
= 0.03 * 0.996 * 0.99 * 0.998 * 0.96 * 0.96 * 0.99 * 0.99 * 0.98 & \\
\approx 0.0259 &
\end{aligned}
$$

If no other action were facilitating picking, then the action idle would have been chosen, see figures 4-7 and 4-6. However, TIAGo++ has two end-effectors, and the hard constraints and components needed for the task are satisfied. The same behaviour tree is constructed for the right arm with a functioning servomotor, resulting in a prior over plans larger than the threshold; hence chosen over other actions, resulting in picking with the right arm, as depicted in the real world experiment in figure 5-11.

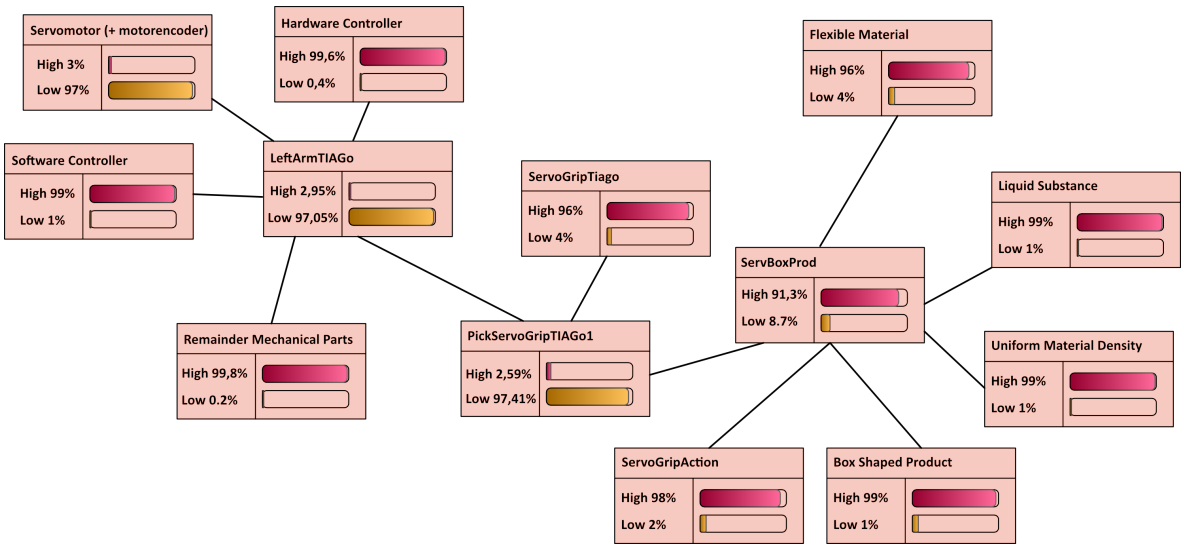


Figure 5-10: Bayesian Network with priors over plans demonstrating the factors influencing the decision making under component failure, with context awareness created through the Airet framework. The Bayesian Network is populated through the Airet ontology. The third case study presented that the servomotor of the left arm failed, causing the picking action utilising the left arm to be less probable for achieving task succession than the functioning right arm. Fault detection & isolation methods are assumed to provide information on the failed components.

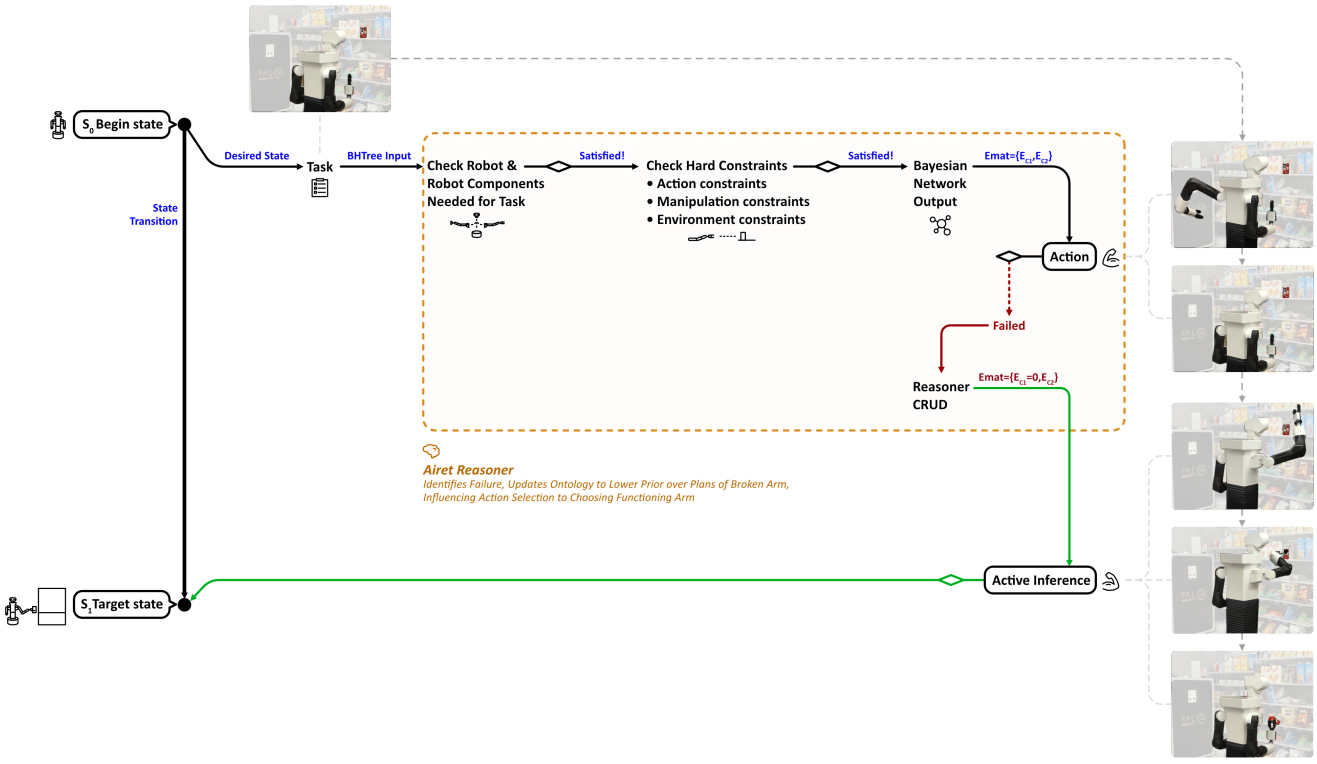


Figure 5-11: Experimental case demonstrating the reasoning capabilities of the Airet framework, under component failure applied on the real TIAGo++ robot in a retail store environment. Component failure is modelled through a signal after seemingly detecting and isolating failure of the servomotor, which could have been obtained from the lower-level controllers. The Airet framework correctly lowers the task succession with the broken left-side arm. The necessary components for task succession with the right-side controllers, hard constraints of the environment, the robot components and object properties are satisfied. The Bayesian Network for picking with the servo-electric end-effector on the right-side arm resulted in this action being chosen as the desired action leading to task succession.

Computational expense of the Airet reasoner

The majority of the computation expense is spent for initializing the Airet server with the Airet ontology, taking a mere 0.147 seconds approximately. This is found through computing with an AMD Ryzen 5, 3600, CPU and 16Gb of random access memory (RAM). Furthermore, this initialization of the ROS service happens upon startup of the robot, hence will not be part of the reasoning time on actions. Loading the Airet ontology into OWLREADY2 using Python takes merely 10,9 miliseconds approximately. The true computational expense for reasoning is found as follows: An example reasoning action being reasoning on picking of a cereal box as in case study 2, see figure 5-5 took approximately 12.7 miliseconds, including construction of the Bayesian Network by the reasoner. This value is taken after the reasoner server and ontology is initialised upon startup. The most computationally expensive reasoning task of all the case studies was case 6, where component failure occurred, see figure 5-11. The reasoner took a total computational expense of 26.6 miliseconds approximately. One can conclude that with this order of computational expense, at run-time, action selection provided through the baseline framework remains unaffected for mobile manipulation in retail with the Airet framework.

5-3 Summary

Ontological reasoning through the Airet framework significantly improves the decision-making through Active Inference, leading to more reliable task execution in comparison to execution with the framework of Pezzato et al. [74]. It is demonstrated through several case studies of ill-defined tasks, preferences due to the situation and controller failure. The user freedom in designing factors influencing decision-making remains large, allowing the user to group factors and deconstruct them into subfactors, as long as consistency is maintained between available actions satisfying the same desired task outcome. Decision-making through Active Inference is made reliable with the Airet framework. The downside of the Airet framework is that priors over plans of (sub-) factors influencing decision-making have to be obtained through expert knowledge, training of neural networks or experimental data on task succession. Learning of these priors can also be obtained through simulation. It is, however, not constraining since these priors over plans are easily reusable, making it primarily a single investment. Furthermore, since the hard constraints are first taken into account before constructing the Bayesian network, more targeted learning or experimentation is achieved. The complexity of which is left free to user design.

Chapter 6

Conclusions & Future Work

A framework called Airet (Active Inference for retail) has been introduced, facilitating cognition for mobile manipulation in a retail environment. More specifically, it satisfies the necessities of deliberate acting mentioned by Ingrand et. Ghallab, which are planning, acting, observing, monitoring, goal reasoning, and learning [47]. This framework consists of a Task and Motion (TAMP) planning module, an ontology containing semantic knowledge needed for mobile manipulation for retail, a reasoning module supporting task planning and decision making, a perception module for obtaining and interpreting percepts of the environment and an execution module containing desired controllers for executing a task.

At the center of this framework lies planning. For this, a novel ontology-centred integration of Active Inference and Ontological Reasoning is proposed to provide failure recovery and improved action selection capabilities for task planning in a retail environment. Situational knowledge is added with respect to a planning alone using Active inference [74], which is supported by reasoning on object (manipulation) properties, task & motion planning, robot capabilities and components, to decide the right action for the situation. This is done to remove the major drawbacks of previous work being alternative actions to achieve the same goal are chosen in a fixed order, lacking context-based adaptation of action selection. This context-based adaptation is crucial when dealing with contingencies like robot component failure.

An ontology is designed to provide semantic knowledge for action planning with Active Inference and knowledge for facilitating reasoning on the environment, including reasoning on the robotic agent's components and capabilities and objects present in a retail environment. The ontology proposed is modelled using the web ontology language OWL and Protégé for visualizing semantic information [1], under the standardised upper ontologies for robotics and automation SUMO, CORA & its extensions CORAX and RPARTS. Furthermore, the ontology utilises knowledge from the planning & manipulation PMK framework and remains compatible with this framework. Standardization is beneficial because it promotes reuse of concepts agreed upon by the scientific community in their respective domains. Furthermore,

[1] https://protege.stanford.edu/

it allows for comparing ontological concept coverage and makes mapping between similar ontologies easier, since the same higher-level primitive concepts would be covered in standardization.

At the heart of the Airet framework lies the reasoner module, which integrates Description Logic reasoners like Pellet & Hermit, with reasoning through Bayesian Networks, the output being feasibility of actions accomplishing a task, and preferences over actions to satisfy tasks given the environment, the task at hand, the robotic agent and unexpected occurring events influencing actions. The reasoning process happens through two stages namely evaluating hard constraints & generating a Bayesian Network. The hard constraints decide action feasibility upon nominal functioning of components, taking into account the robot, the action and the environment. The Bayesian Network considers factors influencing action success, like contingencies such as component failure handling of objects by robot components. The Bayesian network allows for more efficient reasoning & a higher level of modularity & reuse when compared to using a logic based approach solely. Furthermore, it allows for representing uncertainty in actions which is difficult to implement through first-order logic alone. The preferences over actions obtained from the reasoner are sent to the planning module using Active Inference to select preferred actions given the situation at hand.

Communication between packages, including the planning module, controller clients and reasoning module, is provided through the ROS framework. One can find a system overview including these interconnections in section 5-1.

The following case studies are devised and validated through experiments & simulations, to demonstrate the capability of the Airet framework. The results found in these case studies are summarised in table 6-1.

- **Ill-posed Task.**

  The robotic agent has two servo-electric end-effectors. The task is ill-defined, the servo-electric grippers cannot pick up the object due to the object properties (size) causing a gripper constraint (maximum width) to be violated. The first scenario is the baseline [74], which resulted in irrecoverable failure of the task by choosing picking with the robot's right arm. The consequence of this was the object being tipped over on the shelf. The second scenario is through the Airet framework. The Airet reasoner correctly identifies the robot type, its functioning components, the working environment with environmental constraints, the robot and object properties, and failure events to decide the appropriate action for picking a specific object. Since the object properties violate constraints of both end-effectors, only the appropriate action of staying idle is chosen. This is the desired outcome for an infeasible task, guaranteeing safety (no dropping objects or movements of the robot) and allowing the task to be executed by a different robot, having the right components for this task.

- **Choosing Only Suitable Gripper.**

  Different end-effectors are present: one vacuum gripper and one servo-electric gripper. The object's properties combined with the constraints of each end-effector made for a specific action to be preferred over the other, which is determined by the Airet reasoner in combination with the Airet ontology. Picking with the servo-electric gripper was not possible as it is the same task as the previous case. When planned and executed

Case Study	Scenario	Task	Framework	Action	Task Result	Consequence	Reason
1: Ill-posed Task	**1-A**	Pick-up cereal box	**Baseline**	*a_{t}={Pick ServoGripper Rightarm}*	👎 ***RES={Fail}*** - Irrecoverable task **failure**	Safety & Profit **Compromised**	- Box size **exceeding** gripper-**width** - **No reasoner** present to create context awareness
	1-B	Pick-up cereal box	**Airet**	*a_{t}={Stay Idle}*	👎 ***RES={Fail}*** - Task **infeasible** - Action **idle** chosen	Safe	- Box size **exceeding** gripper-**width** - **Airet reasoner** identifies ill-posed **task**
2: Choosing Suitable Gripper	**2-A**	Pick-up cereal box	**Airet**	*a_{t}={Pick VacuumGripper Leftarm}*	👍 ***RES={Success}*** - Task **completed** with vacuum gripper	Task Completion	- **Airet reasoner** identifies box size **exceeding servo gripper-width** - But **satisfying vacuum gripper** manipulation **constraints**
3: Controller Preference	**3-A**	Pick-up yoghurt bottle	**Airet**	*a_{t}={Pick ServoGripper Rightarm}*	👍 ***RES={Success}*** - Task **completed** with preferred gripper	Task Completion	- Robot **components** for action are **present & functioning** - Action, manipulation and environment **constraints** are **satisfied**
4: Component Failure	**4-A**	Pick-up tomato box	**Baseline**	*a_{t}={Pick ServoGripper Leftarm}*	👎 ***RES={Fail}*** - Repeated **failure** - Keeps attempting with broken arm	Safety & Profit **Compromised**	- **No reasoner** present to create context awareness - Hence **no reasoning on failure**
	4-B	Pick-up tomato box	**Airet**	*a_{t}={Stay Idle}*	👍 ***RES={Success}*** - Task **completed** - Adaption is triggered after failure	Task Completion	- **Airet reasoner** identifies **failure**, **updates** ontology to lower prior over plans of broken arm, **influencing action selection** to choosing functioning arm

Table 6-1: Experimental results showing the capabilities of the Airet framework compared to baseline [74]. RES stands for results of task execution with respect to task goal.

with the baseline framework [74], it would be random whether the task succeeds since it depends on the implementation of the baseline framework, which chooses actions in a fixed order. The Airet framework identified this manipulation constraint, and checked the constraints for the vacuum gripper. After these constraints were satisfied, a Bayesian Network was generated to specify likelihood of task success through picking this object with a Vacuum gripper. The Airet reasoner deemed this task feasible with the vacuum gripper solely, hence a preference over plans of this vacuum gripper is given. The result is successful task completion.

- **Controller Preferences Available.**

  The robotic agent has two servo-electric end-effectors. The right servo-electric gripper in this case preferred over the left due to its higher historical reliability, with the left end-effector suffering from more frequent component failure. Other preferences could also be taken into account. For example, if the product is on a certain shelf side, the end-effector on that side is preferred. If planned and executed with the baseline framework [72], when these preferences are not taken into account, it would be random whether the preferred action would be chosen, due to action selection in a fixed order, resulting in task failure being more likely.

- **Component Failure Occurs.**

  The robotic agent has two servo-electric end-effectors. Failure of the left robotic arm occured. The lower-level controllers communicated this to the Airet reasoner. The Airet reasoner reasoned on the failure, updated the Airet ontology of the presence of failure through the CRUD module and then chose the functioning end-effector resulting in succesful task completion. When planned and executed with the baseline framework [74], the robot kept repeating the task with the broken end-effector resulting in task failure and compromising safety, due to repeated unexpected movements made by the robot. This could eventually lead to damaging the robot components, resulting in loss of profit, besides the profit loss caused by the task failure.

Note that a task is considered to have failed when the desired task goal is not achieved. Perception through Active Inference intrinsically checks if the desired state is reached. For picking it would be that the state holding an object is not reached after executing the action, resulting in a higher level of surprise. Task succession is also verified through observing the simulation and real experimental result manually, seeing whether the task goal (e.g. picking of an object) is achieved. *Due to combining hard constraints with the Bayesian Network, the Airet framework produces a deterministic and consistent outcome of the reasoning process.* This has similarities with human thinking in that successful actions given a situation are consistently preferred over less successful actions, after reasoning upon these actions.

Future Work

Reasoning

Reasoning on the feasibility of actions could be enhanced by having more detailed models of environmental constraints and object properties. For example, by integrating a neural network with semantic reasoning, one can relate shelf regions to actions resulting in the most suitable end-effector, robot arm and grasping orientation being chosen for that shelf region. The framework Knowrob has implemented a similar feature that uses detailed information from task execution like recording joint torques, observations of the environment and task goals of successful task executions, to determine if it is safe to pick an object (if an object is in use or not by a human) [5].

Active Inference

Other future work involves defining and reasoning on action models needed for Active Inference, allowing for a more modular specification of actions, pre- and post-conditions. Currently, the generative model of Active inference is hard-coded but could be loaded from the ontology after being reasoned upon by the Airet reasoner, making it dependent on the situation. An example where this is useful is when the uncertainty of the robot about its environment increases due to transitioning to a more unfamiliar part of the retail store. The Airet reasoner could for example reason on events that are occurring in the environment, to adapt the matrix A, which encodes beliefs between hidden states and observable outcomes, to represent observations providing more or less precise information when the agent is in a certain state. This is especially useful in a dynamic environment like retail where uncertainty dictates observations fed into the Active inference algorithm. In sum, the Airet framework would allow for more control on the decision-making by reasoning on the uncertainty and model parameters, varying the generative model of Active inference to counteract this uncertainty.

Task planning

Further work could involve providing a method for online adaptation at the global task level. This could possibly be achieved by integrating a hierarchical task network into the behavior tree, assembling instead of skills behaviour tree branches specifying parts of a plan still feasible after controller failure occurs. This would result in other tasks that are depending on faulty controllers to be omitted from the task plan and the remainder to be executed. An example of this is when a robot with one vacuum and one servo-electric gripper has a list of items to stock. Some of these items could only be manipulated by using the vacuum gripper, as the Airet reasoner would report. If the vacuum gripper fails, all future items to be stocked, relying on a functioning vacuum gripper, would be removed from the task list. The result is that the remaining tasks would still be able to be executed, increasing the profit of a store compared to when stocking actions stop, as well as maintaining customer trust.

Lastly, a Graphical User Interface (GUI) could be developed to facilitate task planning by encoding Active Inference desires in the behavior tree in a graphical more simplistic manner, removing the need for expert knowledge for planning tasks. This could be represented as task blocks being for example "Pick up a tomato can". Due to the modularity of the Airet framework, this would require little to none changes to the Airet ontology and reasoner structure of knowledge. This is especially useful for a retail environment, where many stores are run by people lacking this expertise, and it is infeasible from a monetary cost perspective to always have a technical expert present.

Chapter 7

Appendix

7-1 Appendix A - Motivation Behind KRR: Satisfying Cognition

The definition of cognition as given by Vernon [89] points out a preliminary basis for crucial factors influencing robotic manipulators. Vernon defines cognition as "*the process by which an autonomous system perceives its environment, learns from experience, anticipates the outcome of events, acts to pursue goals, and adapts to changing circumstances*" [89, 91]. These crucial features system architectures should contain for autonomous mental capabilities are discovered by Langley et al [55, 90]. These crucial cognitive capabilities are summarised as follows:

- **Recognition and categorisation**: the ability of a robot to recognise events or situations and categorise them as instances of known patterns. *Adequate minimal level*: Recognise products, shelves, and obstacles in a retail store as well as pick-and-place behaviour of other agents, including humans.
- **Decision-making and choice**: the capability to represent different choices in an understandable matter to a robotic agent and choose between different alternatives. *Adequate minimal level*: Decide which actuators to reach goals, e.g., move, pick, place, change posture and stay idle. Changing posture is relevant when grasping objects with locations at varying heights.
- **Perception and situation assessment**: sensing, perceiving, and interpreting the environment of the robot. Next, the analysis and categorisation of this information into a recognisable flow of events. *Adequate minimal level*: Perceive objects and agents, like a shelf, a basket, a table, a human, a robotic agent, a product sold in a store, and obstacles like shopping carts. Assessing situations includes interpreting recognised patterns like a human moving and pillaging from broken products.
- **Prediction and monitoring**: using the representation of the environment, actions and their effects in order to plan a set of actions for events that have not occurred yet. *Adequate minimal level*: Predict how obstacles like human agents will move with

respect to goal positions such as shelf and stock locations. Predict the consequences of actions, like dropping a bottle, pushing a product on a shelf using the representation of the environment, actions, and their effects to plan a set of actions for events that have not occurred yet.

- **Problem-solving and planning**: The agent should be able to specify appropriate sequences of actions to achieve the desired goals the agent has. Furthermore, when in novel environments, the robot should still be able to specify actions to achieve the desired goal. With contingencies happening, the plan to achieve this goal must be adapted to deal with these unforeseen events, referred to as problem-solving. *Adequate minimal level*: Encode a sequence of tasks to achieve goals as well as motion primitives that lead to an action being satisfied correctly. The former often constitutes moving from stock location, picking objects, moving to shelf, and placing objects, see figure 3-3. Pre- and post-conditions should specify whether tasks such as picking are possible, depending on conditions such as having a free gripper, objects being reachable, and the object weight being lower than the maximum load the controllers can handle. The latter indicates right motion parameters for the base speed of robotic agents, and the grasping method of objects. Also, when contingencies occur, e.g., an actuator fails, e.g., a robot arm, the robot should be capable of recovering and attempting to satisfy the goal. The robot might prioritise other sub-tasks like sorting other products which it is still capable of executing. Other recovery includes missing assertions or predicates in the knowledge base being resolved when encountering a new product.

- **Reasoning and belief maintenance**: ability to form, and keep a representation of beliefs and understand relationships between these beliefs. Furthermore, the adaptation of this representation based on new observations should be included in the capability. *Adequate minimal level*: Describe objects, robotic agent and their properties, extract valuable information for the goal of tasks and adapt the knowledge base to a more up-to-date version. Examples of reasoning are picking up heavy objects with both grippers, using a vacuum gripper for picking up objects wider than a serve gripper, putting objects in shelves where they belong, pushing objects in shelves until the shelf is full, and updating the knowledge base when a product is bought, making the fully stocked shelf re-stockable.

- **Execution and action**: the agent must be able to represent and store motor skills, and translate them into performed commands. *Adequate minimal level*: being able to perform pick, place, move, push, extend and contract actions, understand how these actions affect other obstacles in the world.

- **Interaction and communication**: gaining knowledge from other agents and sending queries back and forth with knowledge requests and discharges. *Adequate minimal level*: Talk to humans and other agents and exchange information. This could be questions regarding product locations, plans the robot should make or task goals the robot should adapt. Information could be knowledge regarding amounts of objects present in the store, expiration dates and locations.

- **Remembering, reflection, and learning**: ability to encode and store results of cognitive tasks for retrieval at a later moment (remembering), analyse these memories to extract useful knowledge (reflection) and generalise the rationale behind the useful

knowledge to situations where this knowledge has a purpose (learning). *Adequate minimal level*: understand which actions cause task goals to fail e.g., dropping an object causes irreversible damage and fails the stocking task. Understand human preferences for product locations based on previous experiences and the importance of these products e.g., panic ensures when baby milk powder is not restocked.

7-2 Appendix B - PMK Classes

- Quantity: [1] Any specification of how many or how much of something there is. Accordingly, there are two subclasses of quantity: number (how many) and physical quantity (how much).
- Quantity Aggregation: A single quantity that represents a set of quantities such as Pose that represents the 3D location of the objects in the environment.
- Attribute: [1] Abstract qualities that can not or are chosen not to be considered as subclasses of Object.
- Artefact Component: [1] Representation of the parts of the workspace object in the world.
- Artefact: [1] An object that is the product of a making.
- Collection: [1] Collections have members like classes, but, unlike classes, they have a position in space-time and members can be added and subtracted without thereby changing the identity of the collection.
- Robot Component: Representation of the parts of the robot in the world.
- Robot: [1] A device in the world that is responsible for executing the tasks.
- Robot Group: [1] A group of robots organized to achieve at least one common goal.
- Measuring Device Component: Representation of the parts of the measuring device (sensor). Measuring Device: [1] Any device whose purpose is to measure a physical quantity.
- Device Group: A group of measuring devices that supply the robot information to achieve one common goal.
- Region: [1] A topographic location. Regions encompass surfaces of objects, imaginary places, and geographic areas.
- Physical Environment: [1] A physical environment is an object that has at least one specific part: a region in which it is located. In addition, a physical environment relates to at least one reference object based on which region is defined.
- Semantic Environment: A physical environment with data (feature) of the artefacts.

- Spatial Context: The circumstances that form the setting for an event that is related to space and which can be fully understood. Specifically, a representation of the world in terms of space.
- Temporal Context: The circumstances that form the setting for an event that is related to time and which can be fully understood. Specifically, a representation of the world in terms of time.
- Situation: The physical object situation in the environment that represents spatially and temporally the relation of the objects to each others.
- Atomic Function: A representation of the processes for motion, manipulation and perception, such as task planners, motion planners or perceptual algorithms. Moreover, it includes primitive actions, pre- and post-conditions related to task planning.
- Sub-task [66] The summarisation of the typical definition is, a representation of a short-term sequence of action.
- Task [66] The summarisation of the typical definition is, a representation of the long-term goals of the symbolic pre-conditions and effects.

Bibliography

[1] "Ieee standard ontologies for robotics and automation," *IEEE Std 1872-2015*, pp. 1–60, 2015.

[2] M. Baioumy, P. Duckworth, B. Lacerda, and N. Hawes, "Active inference for integrated state-estimation, control, and learning," *arXiv preprint arXiv:2005.05894*, 2020.

[3] M. Baioumy, C. Pezzato, C. H. Corbato, N. Hawes, and R. Ferrari, "Towards stochastic fault-tolerant control using precision learning and active inference," *arXiv preprint arXiv:2109.05870*, 2021.

[4] M. Baioumy, C. Pezzato, R. Ferrari, C. H. Corbato, and N. Hawes, "Fault-tolerant control of robot manipulators with sensory faults using unbiased active inference," *arXiv preprint arXiv:2104.01817*, 2021.

[5] G. Bartels, D. Beßler, and M. Beetz, "Episodic memories for safety-aware robots: Knowledge representation and reasoning for robots that safely interact with human co-workers," *KI - Künstliche Intelligenz*, vol. 33, 04 2019.

[6] M. Beetz, D. Beßler, A. Haidu, M. Pomarlan, A. K. Bozcuoğlu, and G. Bartels, "Know rob 2.0—a 2nd generation knowledge processing framework for cognition-enabled robotic agents," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 512–519.

[7] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, and D. Riboni, "A survey of context modelling and reasoning techniques," *Pervasive and mobile computing*, vol. 6, no. 2, pp. 161–180, 2010.

[8] C. L. Buckley, C. S. Kim, S. McGregor, and A. K. Seth, "The free energy principle for action and perception: A mathematical review," *Journal of Mathematical Psychology*, vol. 81, pp. 55–79, 2017.

[9] R. N. Carvalho, K. B. Laskey, and P. C. Costa, "Pr-owl 2.0–bridging the gap to owl semantics," *Uncertainty reasoning for the semantic web II*, pp. 1–18, 2010.

[10] O. Çatal, J. Nauta, T. Verbelen, P. Simoens, and B. Dhoedt, "Bayesian policy selection using active inference," *arXiv preprint arXiv:1904.08149*, 2019.

[11] O. Çatal, T. Verbelen, J. Nauta, C. De Boom, and B. Dhoedt, "Learning perception and planning with deep active inference," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 3952–3956.

[12] Y.-S. Chang, C.-T. Fan, W.-T. Lo, W.-C. Hung, and S.-M. Yuan, "Mobile cloud-based depression diagnosis using an ontology and a bayesian network," *Future Generation Computer Systems*, vol. 43-44, pp. 87–98, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X1400137X

[13] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," *CoRR*, vol. abs/1805.12114, 2018. [Online]. Available: http://arxiv.org/abs/1805.12114

[14] D. Coleman, I. Sucan, S. Chitta, and N. Correll, "Reducing the barrier to entry of complex robotic software: a moveit! case study," *arXiv preprint arXiv:1404.3785*, 2014.

[15] M. Colledanchise and P. Ögren, "How behavior trees modularize hybrid control systems and generalize sequential behavior compositions, the subsumption architecture, and decision trees," *IEEE Transactions on robotics*, vol. 33, no. 2, pp. 372–389, 2016.

[16] ——, *Colledanchise, Michele and Ögren, Petter, Behavior trees in robotics and AI: An introduction*. CRC Press, 2018.

[17] C. H. Corbato, D. Bozhinoski, M. G. Oviedo, G. van der Hoorn, N. H. Garcia, H. Deshpande, J. Tjerngren, and A. Wasowski, "Mros: Runtime adaptation for robot control architectures," *arXiv preprint arXiv:2010.09145*, 2020.

[18] C. H. Corbato, Z. Milosevic, C. Olivares, G. Rodriguez, and C. Rossi, "Meta-control and self-awareness for the ux-1 autonomous underwater robot," in *Iberian Robotics conference*. Springer, 2019, pp. 404–415.

[19] P. Costa and K. Laskey, "Pr-owl: A framework for probabilistic ontologies," vol. 150, 05 2006, pp. 237–249.

[20] M. Cullen, B. Davey, K. J. Friston, and R. J. Moran, "Active inference in openai gym: A paradigm for computational investigations into psychiatric illness," *Biological Psychiatry: Cognitive Neuroscience and Neuroimaging*, vol. 3, no. 9, pp. 809–818, 2018, computational Methods and Modeling in Psychiatry. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2451902218301617

[21] L. Da Costa, T. Parr, N. Sajid, S. Veselic, V. Neacsu, and K. Friston, "Active inference on discrete state-spaces: a synthesis," *Journal of Mathematical Psychology*, vol. 99, p. 102447, 2020.

[22] K. Dentler, R. Cornet, A. Ten Teije, and N. De Keizer, "Comparison of reasoners for large ontologies in the owl 2 el profile," *Semantic Web*, vol. 2, no. 2, pp. 71–87, 2011.

[23] M. Diab, "Knowledge representation and reasoning for perception-based manipulation planning," 2021.

[24] M. Diab, A. Akbari, M. Ud Din, and J. Rosell, "Pmk—a knowledge processing framework for autonomous robotics perception and manipulation," *Sensors*, vol. 19, no. 5, p. 1166, 2019.

[25] Z. Ding, *Bayesowl: A probabilistic framework for uncertainty in semantic web*. University of Maryland, Baltimore County, 2005.

[26] Z. Ding and Y. Peng, "A probabilistic extension to ontology language owl," in *37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the*. IEEE, 2004, pp. 10–pp.

[27] A. Doan, J. Madhavan, P. Domingos, and A. Halevy, "Learning to map between ontologies on the semantic web," in *Proceedings of the 11th international conference on World Wide Web*, 2002, pp. 662–673.

[28] S. Fenz, A. M. Tjoa, and M. Hudec, "Ontology-based generation of bayesian networks," in *2009 International Conference on Complex, Intelligent and Software Intensive Systems*. IEEE, 2009, pp. 712–717.

[29] S. R. Fiorini, J. L. Carbonera, P. Gonçalves, V. A. Jorge, V. F. Rey, T. Haidegger, M. Abel, S. A. Redfield, S. Balakirsky, V. Ragavan, H. Li, C. Schlenoff, and E. Prestes, "Extensions to the core ontology for robotics and automation," *Robotics and Computer-Integrated Manufacturing*, vol. 33, pp. 3–11, 2015, special Issue on Knowledge Driven Robotics and Manufacturing. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736584514000659

[30] J. Fonollosa, E. Neftci, and M. Rabinovich, "Learning of chunking sequences in cognition and behavior," *PLOS Computational Biology*, vol. 11, no. 11, pp. 1–24, 11 2015. [Online]. Available: https://doi.org/10.1371/journal.pcbi.1004592

[31] K. Friston, "Learning and inference in the brain," *Neural Networks*, vol. 16, no. 9, pp. 1325–1352, 2003, neuroinformatics. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608003002454

[32] ——, "Friston, karl, a theory of cortical responses," *Philosophical transactions of the Royal Society B: Biological sciences*, vol. 360, no. 1456, pp. 815–836, 2005.

[33] ——, "Friston, karl, the free-energy principle: a unified brain theory?" *Nature reviews neuroscience*, vol. 11, no. 2, pp. 127–138, 2010.

[34] K. Friston, T. FitzGerald, F. Rigoli, P. Schwartenbeck, and G. Pezzulo, "Active inference: a process theory," *Neural computation*, vol. 29, no. 1, pp. 1–49, 2017.

[35] K. Friston, J. Mattout, and J. Kilner, "Action understanding and active inference," *Biological cybernetics*, vol. 104, no. 1, pp. 137–160, 2011.

[36] K. Friston, C. Thornton, and A. Clark, "Free-energy minimization and the dark-room problem," *Frontiers in Psychology*, vol. 3, p. 130, 2012. [Online]. Available: https://www.frontiersin.org/article/10.3389/fpsyg.2012.00130

[37] K. J. Friston, J. Daunizeau, and S. J. Kiebel, "Reinforcement learning or active inference?" *PloS one*, vol. 4, no. 7, p. e6421, 2009.

[38] K. J. Friston, T. Parr, and B. de Vries, "The graphical brain: belief propagation and active inference," *Network Neuroscience*, vol. 1, no. 4, pp. 381–414, 2017.

[39] K. J. Friston, T. Shiner, T. FitzGerald, J. M. Galea, R. Adams, H. Brown, R. J. Dolan, R. Moran, K. E. Stephan, and S. Bestmann, "Dopamine, affordance and active inference," *PLoS computational biology*, vol. 8, no. 1, p. e1002327, 2012.

[40] K. J. Friston and K. E. Stephan, "Free-energy and the brain," *Synthese*, vol. 159, no. 3, pp. 417–458, 2007.

[41] Goldberger, Gordon, and Greenspan, "An efficient image similarity measure based on approximations of kl-divergence between two gaussian mixtures," in *Proceedings Ninth IEEE International Conference on Computer Vision*, Oct 2003, pp. 487–493 vol.1.

[42] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3389–3396.

[43] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," in *International conference on machine learning*. PMLR, 2016, pp. 2829–2838.

[44] T. Gu, H. K. Pung, D. Q. Zhang, H. K. Pung, and D. Q. Zhang, *A bayesian approach for dealing with uncertain contexts*. na, 2004.

[45] E. M. Helsper and L. C. Van Der Gaag, "Building bayesian networks through ontologies," in *ECAI*, vol. 2002, 2002, p. 15th.

[46] C. Hernández, J. L. Fernández, G. Sánchez-Escribano, J. Bermejo-Alonso, and R. Sanz, "Model-based metacontrol for self-adaptation," in *International Conference on Intelligent Robotics and Applications*. Springer, 2015, pp. 643–654.

[47] F. Ingrand and M. Ghallab, "Deliberation for autonomous robots: A survey," *Artificial Intelligence*, vol. 247, pp. 10–44, 2017, special Issue on AI and Robotics. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370214001350

[48] M. Kalakrishnan, L. Righetti, P. Pastor, and S. Schaal, "Learning force control policies for compliant manipulation," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 4639–4644.

[49] R. Kaplan and K. J. Friston, "Planning and navigation as active inference," *Biological cybernetics*, vol. 112, no. 4, pp. 323–343, 2018.

[50] M. Kirchhoff, T. Parr, E. Palacios, K. Friston, and J. Kiverstein, "The markov blankets of life: autonomy, active inference and the free energy principle," *Journal of The royal society interface*, vol. 15, no. 138, p. 20170792, 2018.

[51] K.-E. Ko and K.-B. Sim, "Development of context aware system based on bayesian network driven context reasoning method and ontology context modeling," in *2008 International Conference on Control, Automation and Systems*, 2008, pp. 2309–2313.

[52] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79 – 86, 1951. [Online]. Available: https://doi.org/10.1214/aoms/1177729694

[53] L. Kunze, N. Hawes, T. Duckett, M. Hanheide, and T. Krajník, "Artificial intelligence for long-term robot autonomy: A survey," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4023–4030, 2018.

[54] M. S. Lacher and G. Groh, "Facilitating the exchange of explicit knowledge through ontology mappings." in *FLAIRS conference*, 2001, pp. 305–309.

[55] P. Langley, J. E. Laird, and S. Rogers, "Cognitive architectures: Research issues and challenges," *Cognitive Systems Research*, vol. 10, no. 2, pp. 141–160, 2009.

[56] K. B. Laskey, R. Haberlin, R. N. Carvalho, and P. C. G. da Costa, "Pr-owl 2 case study: A maritime domain probabilistic ontology." in *STIDS*, 2011, pp. 76–83.

[57] S. Li, S. Chen, and Y. Liu, "A method of emergent event evolution reasoning based on ontology cluster and bayesian network," *IEEE Access*, vol. 7, pp. 15 230–15 238, 2019.

[58] M. Maachou, "Knowledge-based approach for mobile manipulation with active inference, a literature review," 2021, unpublished literature review.

[59] T. Matsumoto and J. Tani, "Goal-directed planning for habituated agents by active inference using a variational recurrent neural network," *Entropy*, vol. 22, no. 5, p. 564, 2020.

[60] B. Millidge, "Deep active inference as variational policy gradients," *Journal of Mathematical Psychology*, vol. 96, p. 102348, 2020.

[61] P. Mitra, N. F. Noy, and A. R. Jaiswal, "Omen: A probabilistic ontology mapping tool," in *International Semantic Web Conference*. Springer, 2005, pp. 537–547.

[62] K. S. Narendra and J. Balakrishnan, "Adaptive control using multiple models," *IEEE transactions on automatic control*, vol. 42, no. 2, pp. 171–187, 1997.

[63] I. Niles and A. Pease, "Towards a standard upper ontology," in *Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001*, 2001, pp. 2–9.

[64] N. F. Noy, "Semantic integration: a survey of ontology-based approaches," *ACM Sigmod Record*, vol. 33, no. 4, pp. 65–70, 2004.

[65] A. Olivares-Alarcos, D. Beßler, A. Khamis, P. Goncalves, M. K. Habib, J. Bermejo-Alonso, M. Barreto, M. Diab, J. Rosell, J. Quintas *et al.*, "A review and comparison of ontology-based approaches to robot autonomy," *The Knowledge Engineering Review*, vol. 34, 2019.

[66] J. I. Olszewska, M. Barreto, J. Bermejo-Alonso, J. Carbonera, A. Chibani, S. Fiorini, P. Goncalves, M. Habib, A. Khamis, A. Olivares, E. P. de Freitas, E. Prestes, S. V. Ragavan, S. Redfield, R. Sanz, B. Spencer, and H. Li, "Ontology for autonomous robotics," in *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2017, pp. 189–194.

[67] J. I. Olszewska, M. Barreto, J. Bermejo-Alonso, J. Carbonera, A. Chibani, S. Fiorini, P. Goncalves, M. Habib, A. Khamis, A. Olivares *et al.*, "Ontology for autonomous robotics," in *2017 26th IEEE international symposium on robot and human interactive communication (RO-MAN)*. IEEE, 2017, pp. 189–194.

[68] R. Pan, Z. Ding, Y. Yu, and Y. Peng, "A bayesian network approach to ontology mapping," in *The Semantic Web – ISWC 2005*, Y. Gil, E. Motta, V. R. Benjamins, and M. A. Musen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 563–577.

[69] D. Paulius and Y. Sun, "A survey of knowledge representation in service robotics," *Robotics and Autonomous Systems*, vol. 118, pp. 13–30, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889018303506

[70] J. Pearl, *Graphical Models for Probabilistic and Causal Reasoning*. Dordrecht: Springer Netherlands, 1998, pp. 367–389. [Online]. Available: https://doi.org/10.1007/978-94-017-1735-9_12

[71] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Networks*, vol. 21, no. 4, pp. 682–697, 2008, robotics and Neuroscience. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608008000701

[72] C. Pezzato, M. Baioumy, C. H. Corbato, N. Hawes, M. Wisse, and R. Ferrari, "Active inference for fault tolerant control of robot manipulators with sensory faults," in *International Workshop on Active Inference*. Springer, 2020, pp. 20–27.

[73] C. Pezzato, R. Ferrari, and C. H. Corbato, "A novel adaptive controller for robot manipulators based on active inference," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2973–2980, 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9000729/?casa_token=OBL93SqhogUAAAAA:6w99NtC-Fk7P0tIiXx6QmTNsWnPS-GKK0OtsJEz-HWgniXwpY0Rtue_qlc5Fe9HQ2vj0ropx7hM

[74] C. Pezzato, C. Hernandez, S. Bonhof, and M. Wisse, "Active inference and behavior trees for reactive action planning and execution in robotics," *arXiv preprint arXiv:2011.09756*, 2020.

[75] A. Rényi *et al.*, "On measures of entropy and information," in *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*. The Regents of the University of California, 1961.

[76] F. Rovida, M. Crosby, D. Holz, A. S. Polydoros, B. Großmann, R. P. Petrick, and V. Krüger, "Skiros—a skill-based robot control platform on top of ros," in *Robot Operating System (ROS)*. Springer, 2017, pp. 121–160.

[77] F. Rovida, B. Grossmann, and V. Krüger, "Extended behavior trees for quick definition of flexible robotic tasks," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 6793–6800.

[78] C. Schlenoff, E. Prestes, R. Madhavan, P. Goncalves, H. Li, S. Balakirsky, T. Kramer, and E. Migueláñez, "An ieee standard ontology for robotics and automation," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 1337–1342.

[79] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[80] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical owl-dl reasoner," *Journal of Web Semantics*, vol. 5, no. 2, pp. 51–53, 2007.

[81] R. Smith, K. Friston, and C. Whyte, "A step-by-step tutorial on active inference and its application to empirical data," 2021.

[82] M. Tenorth and M. Beetz, "Knowrob: A knowledge processing infrastructure for cognition-enabled robots," *The International Journal of Robotics Research*, vol. 32, no. 5, pp. 566–590, 2013.

[83] E. Theodorou, J. Buchli, and S. Schaal, "Reinforcement learning of motor skills in high dimensions: A path integral approach," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 2397–2403.

[84] M. Thosar, S. Zug, A. M. Skaria, and A. Jain, "A review of knowledge bases for service robots in household environments." in *AIC*, 2018, pp. 98–110.

[85] A. Tschantz, M. Baltieri, A. K. Seth, and C. L. Buckley, "Scaling active inference," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[86] A. Tschantz, B. Millidge, A. K. Seth, and C. L. Buckley, "Reinforcement learning through active inference," *arXiv preprint arXiv:2002.12636*, 2020.

[87] K. Ueltzhöffer, "Deep active inference," *Biological cybernetics*, vol. 112, no. 6, pp. 547–573, 2018.

[88] V. Vedral, "The role of relative entropy in quantum information theory," *Rev. Mod. Phys.*, vol. 74, pp. 197–234, Mar 2002. [Online]. Available: https://link.aps.org/doi/10.1103/RevModPhys.74.197

[89] D. Vernon, *Artificial cognitive systems: A primer*. MIT Press, 2014.

[90] D. Vernon, G. Metta, and G. Sandini, "A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 2, pp. 151–180, 2007.

[91] K. Vernon, Ikeuchi, *Computer vision: A reference guide*. Springer Publishing Company, Incorporated, 2014.

[92] W. Wang, G. Zeng, D. Zhang, Y. Huang, Y. Qiu, and X. Wang, "An intelligent ontology and bayesian network based semantic mashup for tourism," in *2008 IEEE Congress on Services-Part I*. IEEE, 2008, pp. 128–135.

[93] Y. Yang and J. Calmet, "Ontobayes: An ontology-driven uncertainty model," in *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*, vol. 1, 2005, pp. 457–463.

[94] D. Yu, K. Yao, H. Su, G. Li, and F. Seide, "Kl-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 7893–7897.

[95] H.-T. Zheng, B.-Y. Kang, and H.-G. Kim, "An ontology-based bayesian network approach for representing uncertainty in clinical practice guidelines," in *Uncertainty reasoning for the semantic web I.* Springer, 2006, pp. 161–173.