Branching Time Active Inference:

the theory and its generality

Théophile Champion TMAC3@KENT.AC.UK

University of Kent, School of Computing

Canterbury CT2 7NZ, United Kingdom

Lancelot Da Costa L.DA-COSTA@IMPERIAL.AC.UK

Imperial College London, Department of Mathematics

London SW7 2AZ, United Kingdom

Wellcome Centre for Human Neuroimaging, University College London

London, WC1N 3AR, United Kingdom

Howard Bowman H.BOWMAN@KENT.AC.UK

University of Birmingham, School of Psychology,

Birmingham B15 2TT, United Kingdom

University of Kent, School of Computing

Canterbury CT2 7NZ, United Kingdom

Marek Grześ M.GRZES@KENT.AC.UK

University of Kent, School of Computing

Canterbury CT2 7NZ, United Kingdom

Editor: TO BE FILLED

Abstract

Over the last 10 to 15 years, active inference has helped to explain various brain mechanisms from habit formation to dopaminergic discharge and even modelling curiosity. However, the current implementations suffer from an exponential (space and time) complexity class when computing the prior over all the possible policies up to the time-horizon. Fountas et al (2020) used Monte Carlo tree search to address this problem, leading to impressive results in two different tasks. In this paper, we present an alternative framework that aims to unify tree search and active inference by casting planning as a structure learning problem. Two tree search algorithms are then presented. The first propagates the expected free energy forward in time (i.e., towards the leaves), while the second propagates it backward (i.e., towards the root). Then, we demonstrate that forward and backward propagations are related to active inference and sophisticated inference, respectively, thereby clarifying the differences between those two planning strategies.

Submitted to Neural Networks

Keywords: Active Inference, Variational Message Passing, Tree Search, Planning, Free Energy Principle

1. Introduction

Active inference is at this point a compelling explanatory approach in cognitive neuroscience, and significant analyses of biologically-realistic implementations in both neural and non-neural communication networks has been conducted. More specifically, active inference extends the free energy principle to generative models with actions (Friston et al, 2016; Da Costa et al, 2020a; Champion et al, 2021b) and can be regarded as a form of planning as inference (Botvinick and Toussaint, 2012). This framework has successfully explained a wide range of neuro-cognitive phenomena, such as habit formation (Friston et al, 2016), Bayesian surprise (Itti and Baldi, 2009), curiosity (Schwartenbeck et al, 2018), and dopaminergic discharges (FitzGerald et al, 2015). It has also been applied to a variety of tasks, such as animal navigation (Fountas et al, 2020), robotic control (Pezzato et al, 2020; Sancaktar et al, 2020), the mountain car problem (Çatal et al, 2020), the game of DOOM (Cullen et al, 2018) and the cart pole problem (Millidge, 2019). Many of those applications require planning several steps into the future in order to be solved successfully. However, as explained in more depth in appendix H, an exhaustive search over all possible sequences of actions will quickly become intractable, i.e., the number of sequences to explore grows exponentially with the time horizon of planning. Figure 1 illustrates this exponential growth. Exploring only a subset of this exponential number of possible sequences using a tree search therefore becomes a compelling and quite natural alternative.

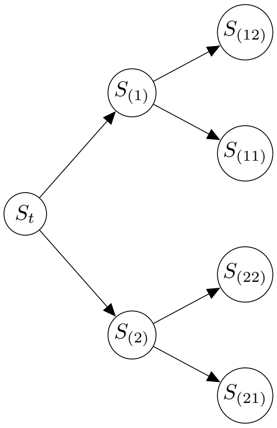


Figure 1: Illustration of all possible policies up to two time steps in the future when $|U| = 2$. The state at the current time step is denoted by S_t. Additionally, each branch of the tree corresponds to a possible policy, and each node S_I is indexed by a multi-index (e.g. $I = (12)$) representing the sequence of actions that led to this state. This should make it clear that for one time step in the future, there are $|U|$ possible policies, after two time steps there are $|U|$ times more policies, and so on until the time-horizon T where there are a total of $|U|^T$ possible policies, i.e., the number of possible policies grows exponentially with the number of time steps for which the agent tries to plan.

But what exactly is active inference? Imagine a basketball player at the top of the key (i.e., the area just below the net) ready to take a shot. Intuitively, active inference sees the world as a collection of external states such as the positions of the net, the player and the ball. The player (or agent) is equipped with sensors (such as the eyes) which allow for measurements of the external states. The player is also able to perform actions in the world such as to perform sudden eye movement or simply unfolding his (or her) arms and legs. Furthermore, it is believed that the agent stores an internal representation of the external states, that we shall refer to as the internal states. Importantly, the external and internal states are separated from each other by the Markov blanket (Kirchhoff et al, 2018), i.e., the sensory information received and actions taken by the agent. In other words, the external states can only modify the internal states indirectly through the observations (also called sensory information) made by the agent, and the internal states can only modify the external states indirectly through the actions taken by the agent.

More formally, active inference builds on a subfield of Bayesian statistics called variational inference (Fox and Roberts, 2012), in which the true posterior distribution is approximated with a variational distribution. This method provides a way to balance the complexity and accuracy of the posterior distribution. The variational approach is only tractable because some statistical dependencies are ignored during the inference process, i.e., the variational distribution is generally assumed to fully factorise, leading to the well known mean-field approximation:

$$Q(X) = \prod_i Q(X_i), \tag{1}$$

where X is the set of all hidden variables of the model, X_i represents the i-th hidden variable, $Q(X)$ is the variational distribution (see below) approximating the posterior $P(X|O)$ where O is the available data, and $Q(X_i)$ is the i-th factor of the variational distribution. In 2005, Winn and Bishop (2005) presented a message-based implementation of variational inference, which has naturally been called variational message passing. And more recently, Champion et al (2021b) realised an active inference scheme using this variational message passing procedure. By combining the Forney factor graph formalism (Forney, 2001) with the method of Winn and Bishop (2005), it becomes possible to create modular implementations of active inference (van de Laar and de Vries, 2019; Cox et al, 2019) that allows users to define their own generative models without the burden of deriving update equations.

However, as just stated, there is a major bottleneck to scaling up the active inference framework: the number of action sequences grows exponentially with the time-horizon (see Appendix H for details). In the reinforcement learning literature, this explosion is frequently handled using Monte Carlo tree search (MCTS) (Silver et al, 2016; Browne et al, 2012; Schrittwieser et al, 2019). This approach has been applied to active inference in several papers (Fountas et al, 2020; Maisto et al, 2021). Fountas et al (2020) chose to modify the original criterion used during the node selection step in MCTS. This step returns the node that needs to be expanded, and the reinforcement learning community uses the upper confidence bound for trees (UCT) introduced by Kocsis and Szepesvári (2006) as a selection criterion:

$$UCT_j = \bar{X}_j + 2C_p\sqrt{\frac{2\ln n}{n_j}}, \tag{2}$$

where n is the number of times the current (parent) node has been explored; n_j stands for the number of times the j-th child node has been explored; $C_p > 0$ is the exploration constant and $\bar{X}_j$ is the average reward received by the j-th child, i.e., the sum of all rewards received by the current node and its descendants divided by n_j. The child node with the largest UCT_j is selected. In their paper, Fountas et al (2020) replaced this selection criterion by:

$$U(s,a) = -\tilde{G}(s,a) + C_{\text{explore}}\ Q(a|s)\ \frac{1}{1+N(s,a)} \tag{3}$$

where $U(s,a)$ indicates the utility of selecting action a in state s; $N(s,a)$ is the number of times that action a was explored in state s; C_{explore} is an exploration constant equivalent to C_p in the UCT criterion; $Q(a|s)$ is a neural network modelling the posterior distribution over actions, which is trained by minimizing the variational free energy, and $\tilde{G}(s,a)$ is an estimator of the expected free energy (EFE). The EFE is computed from the following equation:

$$\begin{aligned}
G(\pi,\tau) = &- \mathbb{E}_{Q(\theta|\pi)Q(s_\tau|\theta,\pi)Q(o_\tau|s_\tau,\theta,\pi)}\Big[\ln P(o_\tau|\pi)\Big] && (4)\\
&+ \mathbb{E}_{Q(\theta|\pi)}\Big[\mathbb{E}_{Q(o_\tau|\theta,\pi)}H(s_\tau|o_\tau,\pi) - H(s_\tau|\pi)\Big] && (5)\\
&+ \mathbb{E}_{Q(\theta|\pi)Q(s_\tau|\theta,\pi)}H(o_\tau|s_\tau,\theta,\pi) - \mathbb{E}_{Q(s_\tau|\pi)}H(o_\tau|s_\tau,\pi), && (6)
\end{aligned}$$

where $H(x|y)$ is the entropy of $p(x|y)$. The computation of the EFE is performed by sampling from three distributions whose parameters are predicted by deep neural networks, i.e., the encoder network modelling $Q(s_\tau)$, the decoder network modelling $P(o_\tau|s_\tau)$ and the transition network modelling $P(s_\tau|s_{\tau-1}, a_{\tau-1})$. Note that Equation (2) was developed by Kocsis and Szepesvári (2006) as a criterion for selecting nodes during planning, such that the selected node minimizes the agent's regret (c.f. Appendix G for additional details). Equation (3) finds its origin in the Predictor Upper Confidence Bound (PUCB) algorithm introduced by Rosin (2010). The idea of the PUCB algorithm is to use contextual information to predict the node to select during planning. Equations (2) and (3) both aim to select the node that minimizes the agent's regret, and can therefore be used interchangeably. However, Equation (3) requires contextual information and a model predicting the node to be selected. Fountas et al (2020) proposed to use the neural network modelling $Q(a|s)$ as a predictor. This has the advantage of making the predictor very flexible, since neural networks are known to be general function approximators, but neural networks are also expensive to train and lack interpretability.

To avoid the additional complexity brought by the predictor, this paper makes use of (2), which arises from the multi-armed bandit literature (Auer et al, 2002). The idea is to minimise the agent's regret to handle the trade-off between exploration and exploitation at the tree-level in an optimal manner.

A major novelty of our paper is to think about tree search as a dynamical expansion of the generative model, where the past and present is modelled as a partially observable Markov decision process (Sondik, 1971) and the future is modelled by a tree-like generative model. Importantly, our agent treats future states and observations as latent variables over which posterior beliefs are computed, and those beliefs encode the uncertainty of our agent over future states. In contrast, Fountas et al (2020) are using a maximum a posteriori (MAP) estimate of the future hidden states, while performing MCTS. Lastly, the posterior beliefs held by our agent are computed using variational message passing as presented in (Champion et al, 2021b). In comparison, Fountas et al (2020) perform amortized inference using an encoder network that predicts the mean and variance of the posterior distribution over latent states. Then, (during planning) a MAP estimate is used as input for the neural network modelling the temporal transition. All those neural networks are trained using gradient descent on the variational free energy.

Overall, the key contribution of our paper is to use MCTS to expand or grow the probabilistic graphical model, treat future states and observations as latent variables, and do inference using variational message passing. Indeed, the definition in (Champion et al, 2021b) of a general message passing procedure for performing active inference makes it possible to construct graphical active inference models in a modular fashion. In turn, this makes it possible to incrementally expand an active inference model as required of our MCTS procedure. It is this message passing procedure that makes our approach possible. To our knowledge, an approach of this kind has never been studied before.

In the following, we first provide the requisite background concerning Forney factor graphs, variational message passing, active inference, and Monte Carlo tree search in Sections 2, 3, 4, and 5, respectively. Next, Section 6 introduces our method that frames planning using a tree as a form of Bayesian model extension. Using terminology from concurrency theory (Bowman, 2005), we call our new formalism *Branching Time Active Inference* (BTAI). In this domain, models of systems based upon sequences of actions (the format of policies) are described as *linear time*, while models based upon tree and even graph structures are called *branching time* (Glabbeek, 1990; van Glabbeek, 1993; Bowman, 2005). Importantly, BTAI does not consider the generative model and the tree as two different objects, instead, BTAI merges those two objects together into a generative model that can be dynamically expanded. For a detailed analysis of the properties of BTAI, the reader is referred to our companion paper (Champion et al, 2021a), which provides an empirical demonstration of the benefits of BTAI over standard active inference (AcI) in the context of a graph navigation task. This companion paper also supplies a theoretical comparison of BTAI and standard AcI based upon a complexity class analysis. Briefly, standard AcI has a space complexity class of $O(|\pi| \times T \times |S|)$, where $|\pi| = |U|^T$ is the number of possible policies, $|U|$ is the number of available actions, T is the time horizon of planning, and $|S|$ is the number of values that the hidden state can take. In contrast, the space complexity class of BTAI is $O([K + t] \times |S|)$, where t is the current (i.e. present) time point, and K is the number of expansions of the tree performed during planning. Importantly, even complex applications such as the game of Go can be solved by expanding only a small number of nodes (Silver et al, 2016; Schrittwieser et al, 2019). Section 6 is followed by Section 7 that explains the connection between our method and the planning strategies used in both active inference and sophisticated inference (Friston et al, 2021). Finally, Section 8 concludes this paper and provides ideas for future research.

2. Forney Factor Graphs

A Forney factor graph (Forney, 2001) uses three kinds of nodes. The nodes representing hidden and observed variables are depicted by white and gray circles, respectively. And the distribution's factors are represented using white squares, which are linked to variable nodes by arrows or lines. Arrows are used to connect factors to their target variable, while lines link factors to their predictors. Figure 2 shows an example of a Forney factor graph corresponding to the following generative model:

$$P(O, S) = P_O(O|S)P_S(S). \tag{7}$$

Generally, factor graphs only describe the model's structure such as the variables and their dependencies, but do not specify the definition of individual factors. For example, the definitions of P_O and P_S are not given by Figure 2, and additional information is required to remove the ambiguity, e.g., $P_S(S) = \mathcal{N}(S; \mu, \sigma)$ clarifies that P_S is a Gaussian distribution.

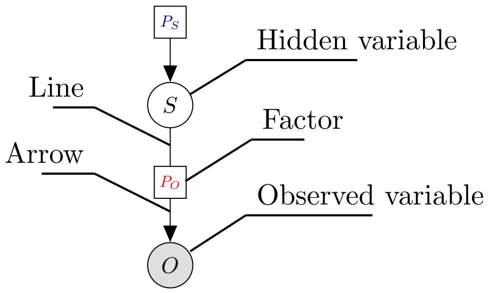


Figure 2: This figure illustrates the Forney factor graph corresponding to the following generative model: $P(O, S) = P_O(O|S)P_S(S)$. The hidden state is represented by a white circle with the variable's name at the center, and the observed variable is depicted similarly but with a gray background. The factors of the generative model are represented by squares with a white background and the factor's name at the center. Finally, arrows connect the factors to their target variable and lines link each factor to its predictor variables.

3. Variational Message Passing

We now build on Forney factor graphs and provide an overview of the method of Winn and Bishop (2005). For more details, see Champion et al (2021b), which provided a complete derivation of the equations presented below from Bayes' theorem.

3.1 Winn and Bishop method

Variational message passing as developed by Winn and Bishop (2005) is an approach for inference based upon the mean-field approximation, which assumes that the posterior fully factorises, i.e.

$$Q(X) = \prod_i Q(X_i), \tag{8}$$

where X is the set of all hidden variables of the model and X_i represents the i-th hidden variable. In this section, we focus on the intuition behind the method, starting with the update equation of an arbitrary hidden state x_k:

$$\ln Q_k^*(x_k) = \langle \ln P(x_k|\mathrm{pa}_k)\rangle_{\sim Q_k} + \sum_{c_j \in \mathrm{ch}_k} \langle \ln P(c_j|x_k, \mathrm{cp}_{kj})\rangle_{\sim Q_k} + C \tag{9}$$

where C is a normalizing constant, and $\langle \cdot \rangle_{\sim Q_k}$ is the expectation over all factors but $Q_k(x_k)$. (9) tells us that the optimal posterior of any hidden states x_k only depends on its Markov blanket, i.e., x_k's parents pa_k, children ch_k and co-parents cp_{kj}. To make (9) more specific, we assume that each random variable of the model is conjugate to its parents (i.e., the posterior has the same functional form as the prior) and is distributed according to a distribution in the exponential family, i.e.,

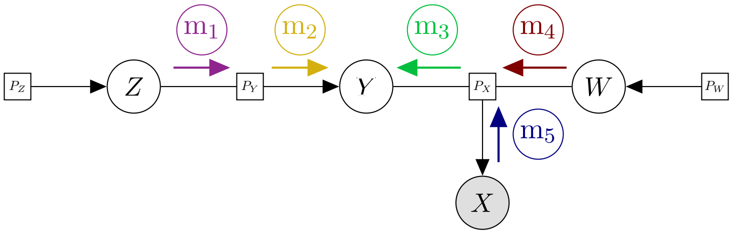
$$\ln P(x_k|\mathrm{pa}_k) = \mu_k(\mathrm{pa}_k) \cdot u_k(x_k) + h_k(x_k) + z_k(\mathrm{pa}_k) \tag{10}$$

where $\mu_k(\mathrm{pa}_k)$, $u_k(x_k)$, $h_k(x_k)$ and $z_k(\mathrm{pa}_k)$ are the parameters, the sufficient statistics, the underlying measure and the log partition, respectively. Under those two assumptions, (9) can be re-written as:

$$Q_k^*(x_k) = \exp\Big\{ \mu_k^* \cdot u_k(x_k) + h_k(x_k) + \mathrm{Const} \Big\} \tag{11}$$

$$\mu_k^* = \tilde{\mu}_k(\{\langle u_i(i)\rangle_{Q_i}\}_{i \in \mathrm{pa}_k}) + \sum_{c_j \in \mathrm{ch}_k} \tilde{\mu}_{j \to k}(\langle u_j(c_j)\rangle_{Q_j}, \{\langle u_l(l)\rangle_{Q_l}\}_{l \in \mathrm{cp}_{kj}}) \tag{12}$$

where $\tilde{\mu}_k$ is a re-parameterization of $\mu_k(pa_k)$ in terms of the expectation of the sufficient statistics of the parents of x_k, and similarly $\tilde{\mu}_{j\to k}$ is a re-parameterization of $\mu_{j\to k}$. Importantly, $u_k(x_k)$ and $h_k(x_k)$ in the optimal posterior (11) are the same as in the prior (10), and only the parameters have changed according to (12).



$$\mu^*_Y = \underbrace{\tilde{\mu}_Y(\overbrace{\langle u_Z(Z)\rangle_{Q_Z}}^{\mathrm{m}_1})}_{\mathrm{m}_2} + \underbrace{\tilde{\mu}_{X\to Y}(\overbrace{\langle u_X(X)\rangle_{Q_X}}^{\mathrm{m}_5}, \overbrace{\langle u_W(W)\rangle_{Q_W}}^{\mathrm{m}_4}}_{\mathrm{m}_3}$$

Figure 3: This figure illustrates the computation of the optimal posterior parameters as a message passing procedure, which requires the transmission of messages from the parent (m_2) and child (m_3) factors. Additionally, the message from the child factor (m_3) requires the computation of messages from the co-parent (m_4) and child (m_5) variables. Also, the message from the parent factor (m_2) requires the computation of a message (m_1) from the parent variable.

To understand the intuition behind (12), let us suppose that we are given the Forney factor graph illustrated in Figure 3 and we wish to compute the posterior of Y. Then, the only parent of Y is Z, the only child of Y is X and the only co-parent of Y with respect to X is W. Therefore, applying (12) to our example leads to the equation presented in Figure 3 whose components can be interpreted as messages. Indeed, each variable (i.e., X, Z and W) sends the expectation of its sufficient statistics (i.e., a message) to the square node in the direction of Y (i.e., either P_X or P_Y). Those messages are then combined using a function (i.e., either $\tilde{\mu}_Y$ or $\tilde{\mu}_{X\to Y}$) whose output (i.e., another set of messages) are summed to obtain the optimal parameters μ^*_Y. The computation of the optimal parameters (12) can then be understood as a message passing procedure. Also, we provide in Appendix C a concrete instance of the approach presented above.

4. Active Inference

This section provides a quick overview of the active inference framework, and Appendix H presents a description of the exponential complexity class that it exhibits. The reader is referred to Appendix F for any notations that might not be explained here. For a more detailed treatment of the active inference framework, we refer the reader to (Champion et al, 2021b; Da Costa et al, 2020a; Smith et al, 2021).

4.1 Generative model

As illustrated in Figure 4, the classic generative model represents the world as a sequence of hidden states generating observations through the matrix $\boldsymbol{A}$. The prior over the initial states is defined by the vector $\boldsymbol{D}$ and the transition between time steps is encoded by a 3-tensor $\boldsymbol{B}$, i.e., one matrix per action. Importantly, the random variable π represents all possible policies up to a given time horizon T and each policy is defined as a sequence of actions, i.e., $\{U_t, ..., U_{T-1}\}$ where $U_\tau \in \{1, ..., |U|\}\ \forall \tau \in \{t, ..., T-1\}$. The prior over the policies is then set such that policies with high probability minimise the EFE, which is defined as follows (Parr and Friston, 2019):

$$\boldsymbol{G}(\pi) \approx \sum_{\tau=t+1}^{T} \Bigg[\underbrace{D_{\text{KL}}[\ \overbrace{Q(O_\tau|\pi)}^{\text{expected outcomes}}\ \ ||\ \ \overbrace{P(O_\tau)}^{\text{prior preferences}}\ ]}_{\text{risk}} + \underbrace{\mathbb{E}_{Q(S_\tau|\pi)}[\text{H}[P(O_\tau|S_\tau)]]}_{\text{ambiguity}} \Bigg] \tag{13}$$

where $Q(O_\tau|\pi) \triangleq \sum_{S_\tau} P(O_\tau|S_\tau)Q(S_\tau|\pi)$, $\text{H}[\cdot]$ is the Shannon entropy, $\boldsymbol{G}$ is a vector containing as many elements as the number of policies, and the i-th element of $\boldsymbol{G}$ represents the cost of the i-th policy. The prior preferences over observations $P(O_\tau)$ represent the (categorical) distribution that the agent wants its observations to be sampled from and is traditionally encoded by the vector $\boldsymbol{C}$. Note that this generalises the concept of reward from reinforcement learning. Indeed, maximising reward can be reformulated as sampling observations from a Dirac delta distribution over reward maximising states (Da Costa et al, 2020b).

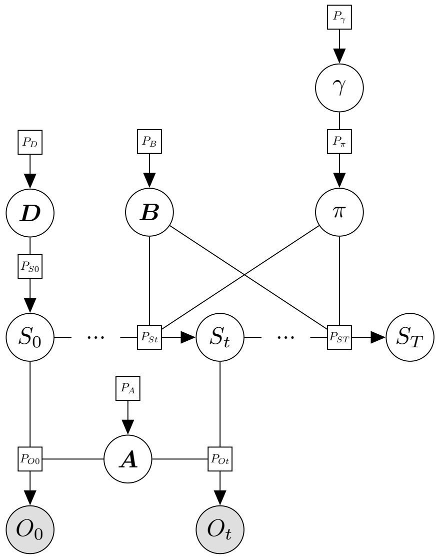


Figure 4: This figure illustrates the Forney factor graph of the entire generative model presented by Friston et al (2016). The probability of the initial states is defined by the vector $\boldsymbol{D}$, and the matrix $\boldsymbol{A}$ defines the probability of the observations given the hidden states. The $\boldsymbol{B}$ matrices define the transition between any successive pair of hidden states. This transition depends on the action performed by the agent, i.e., on the policy π. Furthermore, the prior over the policies has been chosen such that policies minimizing expected free energy are more probable. Finally, the precision parameter γ (which modulates the confidence over which policies to pursue) is distributed according to a gamma distribution.

Lastly, the precision parameter γ has been associated to neuromodulators such as dopamine (FitzGerald et al, 2015; Friston et al, 2013) and can be understood as modulating the confidence over the information afforded by the expected free energy—e.g., smaller values of γ lead to more stochastic decision-making. Finally, the framework allows $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{D}$ to be learned by introducing Dirichlet distributions over the columns of these tensors such that the posterior parameters of $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{D}$ can be reused in a new trial, as parameters of the prior, giving an empirical prior. Finally, the classic generative model is defined as follows:

$$\begin{aligned} P(O_{0:t}, S_{0:T}, \pi, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{D}, \gamma) &= P(\pi|\gamma)P(\gamma)P(\boldsymbol{A})P(\boldsymbol{B})P(S_0|\boldsymbol{D})P(\boldsymbol{D}) \\ &\quad \prod_{\tau=0}^{t} P(O_\tau|S_\tau, \boldsymbol{A}) \prod_{\tau=1}^{T} P(S_\tau|S_{\tau-1}, \pi_{\tau-1}, \boldsymbol{B}) \end{aligned} \tag{14}$$

$$\begin{aligned} &P(\pi|\gamma) = \sigma(-\gamma \boldsymbol{G}) && P(\gamma) = \Gamma(1, \boldsymbol{\beta}) \\ &P(\boldsymbol{A}) = \mathrm{Dir}(\boldsymbol{a}) && P(\boldsymbol{B}) = \mathrm{Dir}(\boldsymbol{b}) \\ &P(S_0|\boldsymbol{D}) = \mathrm{Cat}(\boldsymbol{D}) && P(\boldsymbol{D}) = \mathrm{Dir}(\boldsymbol{d}) \\ &P(O_\tau|S_\tau, \boldsymbol{A}) = \mathrm{Cat}(\boldsymbol{A}) && P(S_\tau|S_{\tau-1}, \pi_{\tau-1}, \boldsymbol{B}) = \mathrm{Cat}(\boldsymbol{B}), \end{aligned}$$

where $\boldsymbol{G}$ is a vector of size $|\pi|$ whose i-th element corresponds to the expected free energy of the i-th policy, $\sigma(\bullet)$ is the softmax function, $\Gamma(\bullet)$, $\mathrm{Cat}(\bullet)$ and $\mathrm{Dir}(\bullet)$ stand for a gamma, categorical and Dirichlet distribution, respectively, $\pi_{\tau-1} \in \{1, ..., |U|\}$ is the action prescribed by policy π at time $\tau-1$, $O_{0:t}$ is the set of (random variables representing) observations between time step 0 and t, and $S_{0:T}$ is the set of (random variables representing) hidden states between time step 0 and T.

4.2 Variational Distribution

The most widely used variational distribution (Da Costa et al, 2020a; Friston et al, 2016) is not fully factorized, i.e., the posterior models the influence of the policy on the hidden states, leading to the following factorization:

$$Q(S_{0:T}, \pi, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{D}, \gamma) = Q(\pi)Q(\boldsymbol{A})Q(\boldsymbol{B})Q(\boldsymbol{D})Q(\gamma) \prod_{\tau=0}^{T} Q(S_\tau|\pi) \tag{15}$$

$$\begin{aligned} &Q(S_\tau|\pi) = \mathrm{Cat}(\hat{\boldsymbol{D}}_\tau) && Q(\pi) = \mathrm{Cat}(\hat{\boldsymbol{\pi}}) \\ &Q(\gamma) = \Gamma(1, \hat{\boldsymbol{\beta}}) && Q(\boldsymbol{D}) = \mathrm{Dir}(\hat{\boldsymbol{d}}) \\ &Q(\boldsymbol{A}) = \mathrm{Dir}(\hat{\boldsymbol{a}}) && Q(\boldsymbol{B}) = \mathrm{Dir}(\hat{\boldsymbol{b}}) \end{aligned}$$

where all variables with a hat correspond to posterior parameters. Notice that the distributions over $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{D}$ remain Dirichlet distributions, and the distributions over γ and S_τ remain a gamma and a categorical distribution, respectively. Only the distribution over π changes from a Boltzmann to a categorical distribution but both are discrete distributions.

Remark 1 *By definition the generative model $P(O_{0:t}, S_{0:T}, \pi, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{D}, \gamma)$ is a joint probability distribution over both the observed ($O_{0:t}$) and latent ($S_{0:T}, \pi, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{D}, \gamma$) variables. However, the goal of the variational distribution $Q(S_{0:T}, \pi, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{D}, \gamma)$ is to approximate the true posterior $P(S_{0:T}, \pi, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{D}, \gamma | O_{0:t})$, which is a distribution over the latent variables only. Thus, the approximate posterior $Q(S_{0:T}, \pi, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{D}, \gamma)$ is also a distribution over the latent variables only, and does not contain the observed variables.*

4.3 Variational Free Energy

By definition, the variational free energy (VFE) is the Kullback-Leibler divergence between the variational distribution and the generative model, i.e.

$$\begin{aligned}
\boldsymbol{F} &= \mathbb{E}_Q[\ln Q(S_{0:T}, \pi, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{D}, \gamma) - \ln P(O_{0:t}, S_{0:T}, \pi, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{D}, \gamma)] &(16)\\
&= D_{\mathrm{KL}}\left[\, Q(x) || \, P(x|o)\right] \; - \ln P(o) &(17)\\
&= \underbrace{D_{\mathrm{KL}}\left[\, Q(x) || \, P(x)\right]}_{\text{complexity}} - \underbrace{\mathbb{E}_{Q(x)}[\ln P(o|x)]}_{\text{accuracy}} &(18)
\end{aligned}$$

where $x = \{S_{0:T}, \pi, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{D}, \gamma\}$ refers to the model's hidden variables, and $o = \{O_{0:t}\}$ refers to the sequence of observations made by the agent. (17) shows that minimising free energy involves moving the variational distribution $Q(x)$ closer to the true posterior $P(x|o)$ in the sense of KL divergence, and that the variational free energy is an upper bound on the negative log evidence. (18) shows the trade-off between complexity and accuracy, where the complexity penalises the divergence of the posterior $Q(x)$ from the prior $P(x)$ and the accuracy scores how likely the observations are given the generative model and current belief of the hidden states.

To fit the variational distribution as closely as possible to the true posterior, the VFE is minimized w.r.t each variational factor, e.g., $Q(\boldsymbol{D})$ and $Q(\boldsymbol{A})$. The minimization process can be solved by iterating the update equations of each factor until convergence of the VFE. More details and intuition about those updates are given by Champion et al (2021b).

4.4 Action selection

In active inference, the simplest strategy to select actions is to compute the evidence for all policies under consideration and then choose the most likely action according to these policies. Mathematically, this amounts to a

Bayesian model average by executing the action with the highest posterior evidence:

$$u_t^* = \arg\max_u \sum_{m=1}^{|\pi|} [u = \pi_t^m] Q(\pi = m) \tag{19}$$

where $|\pi|$ is the number of policies, π_t^m is the action predicted at the current time step by the $m-th$ policy, and $[u = \pi_t^m]$ is an indicator function that equals one if $u = \pi_t^m$ and zero otherwise.

5. Monte Carlo Tree Search

By now, the reader should be familiar with the framework of active inference and how variational message passing combined with the Forney factor graph formalism can be used to compute posterior beliefs. We now turn to the last piece of background required to present the method proposed in this paper: Monte Carlo tree search (MCTS), which is based on the multi-armed bandit literature (c.f. Appendix G for details).

5.1 A four step process

Monte Carlo tree search has been widely used in the reinforcement learning literature as it enables agents to plan efficiently when the evaluation of every possible action sequence is computationally prohibitive (Silver et al, 2016; Browne et al, 2012; Schrittwieser et al, 2019; Fountas et al, 2020). This algorithm essentially builds a tree in which each node corresponds to a future state and each edge represents the action that led to that state. Initially, the tree is only composed of a root node corresponding to the current state. From here, MCTS is a four step process. First, a node is selected according to a criterion such as the upper confidence bound for trees (UCT):

$$UCT_j = \bar{X}_j + 2C_p\sqrt{\frac{2\ln n}{n_j}}, \tag{20}$$

where n is the number of times the current (parent) node has been explored, n_j stands for the number of times the j-th child node has been explored, $C_p > 0$ is the exploration constant and $\bar{X}_j$ is the average reward received by the j-th child. Note, if the rewards are in $[0, 1]$, then $C_p = \frac{1}{\sqrt{2}}$ is known to satisfy the Hoeffding inequality (Browne et al, 2012) and the UCT criterion reduces to:

$$UCT_j = \bar{X}_j + 2\sqrt{\frac{\ln n}{n_j}}. \tag{21}$$

Importantly, the UCT aims to explore highly rewarding paths (exploitation in first term), while also visiting rarely explored regions (exploration in second term).

As shown in Figure 5, this criterion is first used at the root level leading to the selection of a node from the root's children. Then, it is used at the level of the root's children, and so on until a leaf node is reached. As explained by Kocsis and Szepesvári (2006), UCT is a direct application of the $UCB1$ criterion to trees, where at each level, the allocation strategy must pick a node that is expected to lead to the highest reward, and "picking the i-th node" can be seen as the i-th action of a multi-armed bandit problem. Once a leaf node has been selected, an expansion step is performed by sampling an action from a distribution and adding the node corresponding to this action as a child of the leaf node, i.e., the leaf node is expanded.

The third step consists of performing virtual rollouts into the future to estimate the average future reward obtained from the state corresponding to the newly expanded node. Finally, during the back-propagation step, the average reward obtained from the newly expanded state is used to re-evaluate the average quality of all its ancestors, and the visit counts of all nodes (in the branch explored) are increased. Iterating this four-step process until the time budget has been spent gives a fairly good estimate of the best action to perform next. Figure 5 summarises the MCTS procedure. In the next section, we present our approach and show how MCTS can be fused to active inference by performing a dynamical expansion of the generative model.

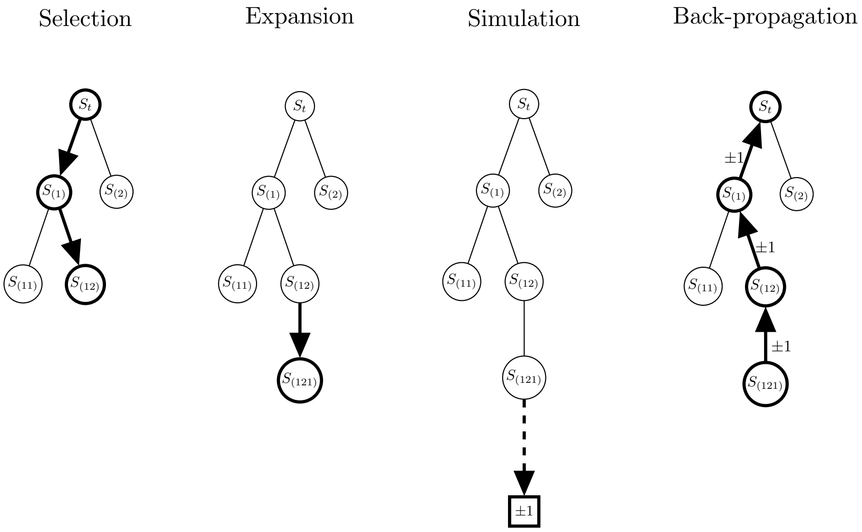


Figure 5: This figure illustrates the MCTS algorithm as a four step process. First, we start at the node representing the current state S_t and select a node based on the UCT criterion until a leaf node is reached. Second, the tree is expanded to a new node by taking a virtual action from the selected node. Third, the value of this action is estimated by simulating the expected reward following that action. In the simplest version of MCTS, simulations are run until a terminal state is reached, e.g., until the game ends in Go or Chess. Fourth, the expected value is back-propagated to the new node and all of its ancestor nodes. The multi-indices in curly brackets denote action sequences taken from the root node, indicating the current state of the environment.

6. Branching Time Active Inference (BTAI)

In this section, we present a novel active inference agent that frames planning using a tree as a form of Bayesian model extension. Using terminology from concurrency theory (Bowman, 2005), we call our new formalism, *Branching Time Active Inference* (BTAI). In this domain, models of systems based upon sequences of actions (the format of policies) are described as *linear time*, while models based upon tree and even graph structures are called *branch-*

ing time (Glabbeek, 1990; van Glabbeek, 1993; Bowman, 2005). Importantly, we do not consider the generative model and the tree as two different objects. Instead, we merge those two objects together into a generative model that can be dynamically expanded.

Figure 6 illustrates an example of such a model, where for the sake of simplicity, we assume that the matrices $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{D}$ are given to the agent. Furthermore, the random variable representing the policies has been replaced by random variables representing actions and the precision parameter γ has been removed, which is a common design choice (Fountas et al, 2020). Additionally, we follow Parr and Friston (2019) by viewing future observations as latent random variables. Finally, note that the transition between two consecutive hidden states in the future ($S_{I \setminus last}$ and S_I where I is a multi-index) will only depend on the matrix $\bar{\boldsymbol{B}}_I = \bar{\boldsymbol{B}}(\bullet, \bullet, I_{\text{last}})$, i.e., the matrix corresponding to action I_{last} that led to the transition from $S_{I \setminus last}$ to S_I. The reader is referred to Table 1 for the definition of $\bar{\boldsymbol{B}}$ and more details about multi-indices can be found in Appendix F.

6.1 Prior, Posterior and Target distributions

Since the generative model is fairly different from the standard model, we state here its formal definition:

$$P(O_{0:t}, S_{0:t}, U_{0:t-1}, O_{\mathbb{I}_t}, S_{\mathbb{I}_t}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{D}, \boldsymbol{\Theta}_{0:t-1}) = P(S_0|\boldsymbol{D})P(\boldsymbol{A})P(\boldsymbol{B})P(\boldsymbol{D}) \prod_{\tau=0}^{t} P(O_\tau|S_\tau, \boldsymbol{A})$$
$$\prod_{\tau=0}^{t-1} P(U_\tau|\boldsymbol{\Theta}_\tau)P(\boldsymbol{\Theta}_\tau) \prod_{\tau=1}^{t} P(S_\tau|S_{\tau-1}, U_{\tau-1}, \boldsymbol{B}) \prod_{I \in \mathbb{I}_t} P(O_I|S_I)P(S_I|S_{I \setminus last}) \tag{22}$$

where $\mathbb{I}_t$ is the set of all non-empty multi-indices already expanded by the tree search from the current state S_t, the second product (τ from 0 to $t-1$) models the uncertainty over action, reflecting the focus on actions rather than policies, and $S_{I \setminus last}$ is the parent state of S_I. Intuitively, the product over all $I \in \mathbb{I}_t$ models the future, while the rest of the above equation models the past and present. Additionally, we need to define the individual factors:

$$P(S_0|\boldsymbol{D}) = \text{Cat}(\boldsymbol{D}) \qquad P(U_\tau|\boldsymbol{\Theta}_\tau) = \text{Cat}(\boldsymbol{\Theta}_\tau)$$
$$P(O_\tau|S_\tau, \boldsymbol{A}) = \text{Cat}(\boldsymbol{A}) \qquad P(O_I|S_I) = \text{Cat}(\bar{\boldsymbol{A}})$$
$$P(S_\tau|S_{\tau-1}, U_{\tau-1}, \boldsymbol{B}) = \text{Cat}(\boldsymbol{B}) \qquad P(S_I|S_{I \setminus last}) = \text{Cat}(\bar{\boldsymbol{B}}_I)$$
$$P(\boldsymbol{D}) = \text{Dir}(\boldsymbol{d}) \qquad P(\boldsymbol{\Theta}_\tau) = \text{Dir}(\boldsymbol{\theta}_\tau)$$
$$P(\boldsymbol{A}) = \text{Dir}(\boldsymbol{a}) \qquad P(\boldsymbol{B}) = \text{Dir}(\boldsymbol{b})$$

where $\bar{\boldsymbol{A}}$ and $\bar{\boldsymbol{B}}$ are defined in Table 1, $\bar{\boldsymbol{B}}_I = \bar{\boldsymbol{B}}(\bullet, \bullet, I_{\text{last}})$ is the matrix corresponding to I_{last} and I_{last} is the last index of the multi-index I, i.e., the last action that led to S_I. Importantly, $\bar{\boldsymbol{A}}$ and $\bar{\boldsymbol{B}}$ should not be confused with $\mathring{\boldsymbol{A}}$ and $\mathring{\boldsymbol{B}}$, $\bar{\boldsymbol{A}}$ is the expectation of $\boldsymbol{A}$ w.r.t. $Q(\boldsymbol{A})$, while $\mathring{\boldsymbol{A}}$ is the expectation of the logarithm of $\boldsymbol{A}$ w.r.t. $Q(\boldsymbol{A})$.

We now turn to the definition of the variational posterior. Under the mean-field approximation:

$$Q(S_{0:t}, U_{0:t-1}, O_{\mathbb{I}_t}, S_{\mathbb{I}_t}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{D}, \boldsymbol{\Theta}_{0:t-1}) = \\ Q(\boldsymbol{A})Q(\boldsymbol{B})Q(\boldsymbol{D}) \prod_{\tau=0}^{t-1} Q(U_\tau)Q(\boldsymbol{\Theta}_\tau) \prod_{\tau=0}^{t} Q(S_\tau) \prod_{I \in \mathbb{I}_t} Q(O_I)Q(S_I) \tag{23}$$

where the individual factors are defined as:

$$\begin{aligned} &Q(S_\tau) = \text{Cat}(\hat{\boldsymbol{D}}_\tau) && Q(U_\tau) = \text{Cat}(\hat{\boldsymbol{\Theta}}_\tau) \\ &Q(O_I) = \text{Cat}(\hat{\boldsymbol{E}}_I) && Q(S_I) = \text{Cat}(\hat{\boldsymbol{D}}_I) \\ &Q(\boldsymbol{D}) = \text{Dir}(\hat{\boldsymbol{d}}) && Q(\boldsymbol{\Theta}_\tau) = \text{Dir}(\hat{\boldsymbol{\theta}}_\tau) \\ &Q(\boldsymbol{A}) = \text{Dir}(\hat{\boldsymbol{a}}) && Q(\boldsymbol{B}) = \text{Dir}(\hat{\boldsymbol{b}}), \end{aligned}$$

where $\hat{\boldsymbol{D}}_\tau$, $\hat{\boldsymbol{\Theta}}_\tau$, $\hat{\boldsymbol{E}}_I$, $\hat{\boldsymbol{D}}_I$, $\hat{\boldsymbol{d}}$, $\hat{\boldsymbol{\theta}}_\tau$, $\hat{\boldsymbol{a}}$ and $\hat{\boldsymbol{b}}$ are the parameters of the factors $Q(S_\tau)$, $Q(U_\tau)$, $Q(O_I)$, $Q(S_I)$, $Q(\boldsymbol{D})$, $Q(\boldsymbol{\Theta}_\tau)$, $Q(\boldsymbol{A})$ and $Q(\boldsymbol{B})$, respectively. Importantly, O_I appears in the variational distribution because observations in the future are treated as hidden variables.

Finally, we follow Millidge et al (2021) in assuming that the agent aims to minimise the KL divergence between the approximate posterior depicting the state of the environment and a target (desired) distribution. Therefore, our framework allows for the specification of prior preferences over both future hidden states and future observations:

$$V(O_{\mathbb{I}_t}, S_{\mathbb{I}_t}) = \prod_{I \in \mathbb{I}_t} V(O_I)V(S_I) \tag{24}$$

where the individual factors are defined as:

$$V(O_I) = \text{Cat}(\boldsymbol{C}_O), \qquad V(S_I) = \text{Cat}(\boldsymbol{C}_S). \tag{25}$$

Importantly, by specifying the value of future observations and states, $\boldsymbol{C}_O$ and $\boldsymbol{C}_S$ play a similar role to the vector $\boldsymbol{C}$ in active inference, i.e., they specify which observations and hidden states are rewarding.

To sum up, this framework is defined using three distributions: the prior defines the agent's beliefs before sampling any observation; the posterior is an updated version of the prior which takes into account past observations made by the agent; finally, the target distribution encodes the agent's prior preferences in terms of future observations and hidden states.

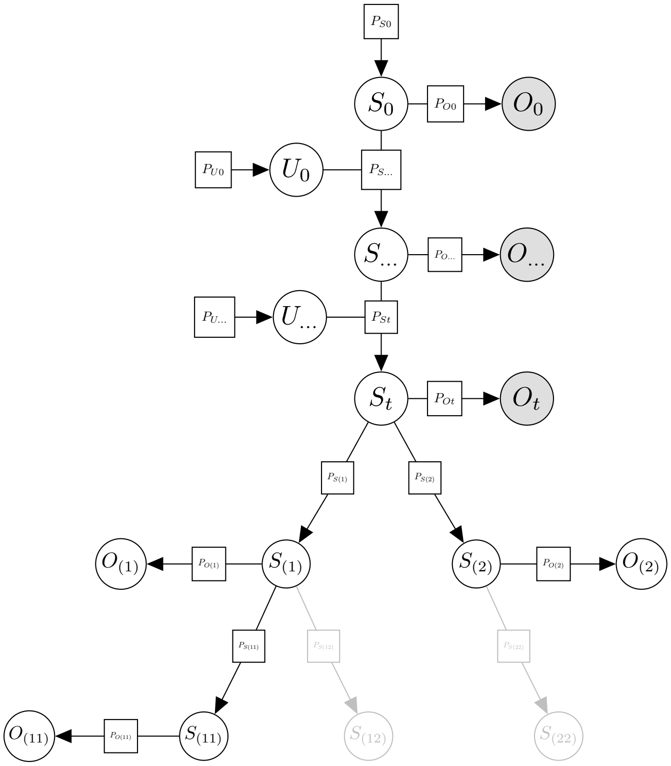


Figure 6: This figure illustrates the new expandable generative model allowing planning under active inference. The future is now a tree like generative model whose branches correspond to the policies considered by the agent. As we will see, these branches can be dynamically expanded during planning. Here, the nodes in light gray represent possible expansions of the current generative model. For the sake of clarity, the random tensor $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{\Theta}_\tau$ and $\boldsymbol{D}$ are not illustrated, i.e., Dirichlet priors over those random tensors are not shown.

6.2 Bayesian belief updates

In this section, we focus on the set of update equations used to perform approximate Bayesian inference. These update equations rely on variational message passing as presented in Section 3, see Champion et al (2021b) as well as Winn and Bishop (2005) for details. A key strength of the message passing approach is the capacity to derive and implement these updates within an automatic and modular toolbox (van de Laar and de Vries, 2019; Cox et al, 2019), which in a way similar to automatic differentiation alleviates the final user from the burden of manually deriving complex update equations for each new generative model. To simplify our notation, we use two operators $\otimes$ and $\odot$ that we call generalized outer and inner product, respectively. The generalized outer product creates an N dimensional tensor from N vectors, while the generalized inner product performs a weighted average over one dimension of an N dimensional array, cf Appendix A for details. Using these notations, the first set of update equations are given by:

$$Q^*(\boldsymbol{D}) = \mathrm{Dir}\big(\hat{\boldsymbol{d}}\big) \quad \text{where} \quad \hat{\boldsymbol{d}} = \boldsymbol{d} + \hat{\boldsymbol{D}}_0 \tag{26}$$

$$Q^*(\boldsymbol{A}) = \mathrm{Dir}\big(\hat{\boldsymbol{a}}\big) \quad \text{where} \quad \hat{\boldsymbol{a}} = \boldsymbol{a} + \sum_{\tau=0}^{t} \otimes\Big[\hat{\boldsymbol{D}}_\tau, \boldsymbol{o}_\tau\Big] \tag{27}$$

$$Q^*(\boldsymbol{B}) = \mathrm{Dir}\big(\hat{\boldsymbol{b}}\big) \quad \text{where} \quad \hat{\boldsymbol{b}} = \boldsymbol{b} + \sum_{\tau=1}^{t} \otimes\Big[\hat{\boldsymbol{D}}_{\tau-1}, \hat{\boldsymbol{\Theta}}_{\tau-1}, \hat{\boldsymbol{D}}_\tau\Big] \tag{28}$$

$$Q^*(\boldsymbol{\Theta}_\tau) = \mathrm{Dir}\big(\hat{\boldsymbol{\theta}}_\tau\big) \quad \text{where} \quad \hat{\boldsymbol{\theta}}_\tau = \boldsymbol{\theta}_\tau + \hat{\boldsymbol{\Theta}}_\tau \tag{29}$$

where $\boldsymbol{o}_\tau$ is the observation made at time τ. Furthermore, this first set of equations count (probabilistically) the number of times, an initial hidden state has been observed, an action has been performed, a state has generated a particular observation or an action has led to the transition between two consecutive hidden states. For example, the posterior parameters $\hat{\boldsymbol{a}}$ are computed by adding $\sum_{\tau=0}^{t} \otimes[\hat{\boldsymbol{D}}_{\boldsymbol{\tau}}, \boldsymbol{o}_\tau]$ (i.e., the number of times a state-observation pair has been observed during this trial) to the prior parameters $\boldsymbol{a}$ (i.e., the number of times this same pair has been observed during previous trials). The equations for belief updates are given by:

$$Q^*(O_I) = \sigma\big(\mathring{\boldsymbol{A}} \odot \hat{\boldsymbol{D}}_I\big) \tag{30}$$

$$Q^*(S_I) = \sigma\Big(\mathring{\boldsymbol{A}} \odot \hat{\boldsymbol{E}}_I + \mathring{\boldsymbol{B}}_I \odot \hat{\boldsymbol{D}}_{I\setminus last} + \sum_{J \in \mathrm{ch}_I} \mathring{\boldsymbol{B}}_J \odot \hat{\boldsymbol{D}}_J\Big) \tag{31}$$

$$Q^*(U_\tau) = \sigma\big(\mathring{\boldsymbol{\Theta}} + \mathring{\boldsymbol{B}} \odot [\hat{\boldsymbol{D}}_\tau, \hat{\boldsymbol{D}}_{\tau+1}]\big) \tag{32}$$

$$\begin{aligned} Q^*(S_\tau) = \sigma\Big(&[\tau = 0]\mathring{\boldsymbol{D}}_\tau \; + \; [\tau \neq 0]\mathring{\boldsymbol{B}} \odot [\hat{\boldsymbol{D}}_{\tau-1}, \hat{\boldsymbol{\Theta}}_{\tau-1}] \\ &+\mathring{\boldsymbol{A}} \odot \boldsymbol{o}_\tau \\ &+[\tau = t] \sum_{J \in \mathrm{ch}_t} \mathring{\boldsymbol{B}}_J \odot \hat{\boldsymbol{D}}_J \; + \; [\tau \neq t]\mathring{\boldsymbol{B}} \odot [\hat{\boldsymbol{D}}_{\tau+1}, \hat{\boldsymbol{\Theta}}_\tau]\Big) \end{aligned} \tag{33}$$

where $\sigma(\bullet)$ is the softmax function, ch_t are the children (states) of the current states S_t, ch_I are the children (states) of the states S_I, [predicate] is an indicator function returning one if the predicate is true and zero otherwise, and the definition of $\mathring{\boldsymbol{A}}$, $\mathring{\boldsymbol{B}}$, $\mathring{\boldsymbol{D}}$ and $\mathring{\boldsymbol{\Theta}}_\tau$ are given in Table 1. Note that thanks to the operators $\otimes$ and $\odot$, the perception (i.e., state-estimation) equations can be intuitively understood as a sum of messages, where each message from a factor to a variable is the average over all dimensions except the dimension of the variable, e.g., the message $(\mathring{\boldsymbol{A}} \odot \boldsymbol{o}_0)$ from P_{o_0} to S_0 is the vector obtained by weigthing the rows of $\mathring{\boldsymbol{A}}$ by the elements of $\boldsymbol{o}_0$. Importantly, the above update equations are almost identical to the ones used in standard active inference, and thus can be implemented efficiently. Indeed, most of the computation required is about addition of matrices $O(n^2)$ and multiplication of matrices $O(n^3)$, or their higher dimensional counterparts.

Notation	Meaning
$\langle f(X) \rangle_{P_X} \triangleq \mathbb{E}_{P_X}[f(X)]$	The expectation of $f(X)$ over P_X
$\psi(\bullet)$	The digamma function
$\mathring{\boldsymbol{\Theta}}_\tau(i) = \langle \ln \boldsymbol{\Theta}_\tau(i) \rangle_{Q_{\Theta_\tau}} = \psi\big(\hat{\boldsymbol{\theta}}_\tau(i)\big) - \psi\big(\sum_k \hat{\boldsymbol{\theta}}_\tau(k)\big)$	The expected logarithm of $\boldsymbol{\Theta}_\tau$
$\mathring{\boldsymbol{D}}(i) = \langle \ln \boldsymbol{D}(i) \rangle_{Q_D} = \psi\big(\hat{\boldsymbol{d}}(i)\big) - \psi\big(\sum_k \hat{\boldsymbol{d}}(k)\big)$	The expected logarithm of $\boldsymbol{D}$
$\mathring{\boldsymbol{A}}(i,j) = \langle \ln \boldsymbol{A}(i,j) \rangle_{Q_A} = \psi\big(\hat{\boldsymbol{a}}(i,j)\big) - \psi\big(\sum_k \hat{\boldsymbol{a}}(k,j)\big)$	The expected logarithm of $\boldsymbol{A}$
$\mathring{\boldsymbol{B}}(i,j,u) = \langle \ln \boldsymbol{B}(i,j,u) \rangle_{Q_B} = \psi\big(\hat{\boldsymbol{b}}(i,j,u)\big) - \psi\big(\sum_k \hat{\boldsymbol{b}}(k,j,u)\big)$	The expected logarithm of $\boldsymbol{B}$
$\bar{\boldsymbol{A}}(i,j) = \langle \boldsymbol{A}(i,j) \rangle_{Q_A} = \frac{\hat{\boldsymbol{a}}(i,j)}{\sum_k \hat{\boldsymbol{a}}(k,j)}$	The expectation of $\boldsymbol{A}$
$\bar{\boldsymbol{B}}(i,j,u) = \langle \boldsymbol{B}(i,j,u) \rangle_{Q_B} = \frac{\hat{\boldsymbol{b}}(i,j,u)}{\sum_k \hat{\boldsymbol{b}}(k,j,u)}$	The expectation of $\boldsymbol{B}$

Table 1: Update equations notation. Note that Appendix D provides a proof for $\boldsymbol{D}$.

6.3 Planning as structure learning

In this section, we frame planning as a form of structure learning where the structure of the generative model is modified dynamically. This method is greatly inspired by the Monte Carlo tree search literature, c.f., Section 5 for details.

6.3.1 Selection of the node to be expanded

The first step of planning is to select a node to be expanded. The selection process starts at the root node, if the root node still has unexplored children, then one of them is selected. Otherwise, the child node maximizing the *UCT* criterion, where the average reward is replaced by minus the average EFE, is selected, i.e., the selected node maximises:

$$UCT_J = \underbrace{-\bar{g}_J}_{\text{exploitation}} + \underbrace{C_p\sqrt{\frac{\ln n}{n_J}}}_{\text{exploration}}, \tag{34}$$

where J is a multi-index, n is the number of times the root node has been visited, n_J is the number of times the child corresponding to the multi-index J was selected, and $\bar{g}_J$ is the average cost received when selecting the child S_J. The *UCT* criterion can be understood as a trade-off between exploitation and exploration at the tree level, which is different to the exploitation and exploration dilemma at the model level. This dilemma is handled by the EFE. Also, the notion of cost in the above equation can be defined in many ways and will be the subject of Section 6.3.3. For our purposes, the cost will be equal, or similar, to the expected free energy, which means that the expected free energy drives structure learning. When a root's child is selected, it becomes the new root in the above procedure, which is iterated until a leaf node is reached.

6.3.2 Dynamical expansion of the generative model

Let $S_{I \setminus last}$ denotes the leaf node selected for expansion. When $S_{I \setminus last}$ has been selected, the structure of the generative model needs to be modified by expanding all possible actions from that node. For each action, we expand the generative model by adding a future hidden state whose prior distribution is given by

$$P(S_I | S_{I \setminus last}) = \text{Cat}(\bar{\boldsymbol{B}}_I), \tag{35}$$

where $\bar{\boldsymbol{B}}_I$ is the matrix corresponding to the last action that led to S_I. Finally, we expand the (future) observation associated with the new hidden state S_I, whose distribution is:

$$P(O_I | S_I) = \text{Cat}(\bar{\boldsymbol{A}}). \tag{36}$$

To sum up, the expansion step is adding two random variables (S_I and O_I) to the generative model, i.e. the generative model becomes bigger, and I is added to the set of all non-empty multi-indices already expanded by the tree search ($\mathbb{I}_t$). The prior distributions over those newly added random variables (i.e. S_I and O_I) are defined using the matrices $\bar{\boldsymbol{B}}_I$ and $\bar{\boldsymbol{A}}$, which effectively predict the future states and observations. After the expansion step, the posterior distribution over S_I and O_I needs to be computed. At least two kinds of inference strategies can be used. The first—global inference—performs variational message passing over the entire generative model, while the second—local inference—only iterates the update equations of the newly expanded nodes, i.e., S_I and O_I, until convergence to the variational free energy minimum.

6.3.3 Cost evaluation of the expanded nodes

After expanding the model structure, we need to compute the cost of the newly expanded node S_I. As explained in Section 6.3.1, the cost of S_I will influence the probability of expanding S_I during future planning iterations. In active inference, the classic objective of planning is the expected free energy as defined in Section 4.1, i.e.,

$$g_I^{classic} \triangleq D_{\text{KL}}[Q(O_I)||V(O_I)] \; + \; \mathbb{E}_{Q(S_I)}[\text{H}[P(O_I|S_I)]] \tag{37}$$

where $g_I^{classic}$ trades off risk (first summand) and ambiguity (second summand). Alternatively, one could follow Section 5 of Millidge et al (2021) and define the cost of S_I using the free energy of the expected future:

$$g_I^{feef} = D_{\mathrm{KL}} \left[Q(O_I, S_I) || V(O_I, S_I) \right] \tag{38}$$

where $V(O_I, S_I)$ is the target distribution over states and observations. The target distribution $V(O_I, S_I)$ generalises the $\boldsymbol{C}$ matrix in Friston's model by specifying prior preferences over both future observations and future states. Also, this formulation of the cost speaks to the notion of KL divergence minimization proposed by Hafner et al (2020).

Furthermore, due to the mean-field approximation of the posterior (23) and the factorised form of the target distribution (24), the expression of the cost simplifies to

$$g_I^{pcost} \triangleq D_{\mathrm{KL}} \left[Q(S_I) || V(S_I) \right] + D_{\mathrm{KL}} \left[Q(O_I) || V(O_I) \right]. \tag{39}$$

Intuitively, the pure cost (g_I^{pcost}) measures how different the predicted future hidden states and observations are from desired states and observations. In future research, it might be interesting to compare the performance and behaviour of g_I^{pcost}, g_I^{feef} and $g_I^{classic}$ empirically and theoretically.

Note that in MCTS, the evaluation of a node's quality is done by performing virtual roll-outs, while g_I^{pcost}, g_I^{feef} and $g_I^{classic}$ are not. If we let g_I be any of those criteria, then we can improve our estimate of the cost, by computing $g_I^{average}$, i.e., the average cost over N roll-outs of size K. Algorithm 1 presents the pseudo code used to estimate $g_I^{average}$.

Algorithm 1: Estimation of $g_I^{average}$

```
**Input:** $N$ the number of virtual roll-outs, $K$ the maximal length of each roll-out.
$g_I^{average} \leftarrow 0$ ;                                    // Initialize roll-out estimate to zero
**repeat** $N$ **times**
    $g_I^{rollout} \leftarrow g_I$ ;                                  // Initial cost equals cost of $S_I$
    **for** $i \leftarrow 1$ **to** $K$ **do**
        sample a random action $U_i$ uniformly from the set of unexplored actions;
        perform the expansion of the current node using $U_i$ (Section 6.3.2);
        perform inference on the newly expanded nodes (Section 6.2);
        $g_I^{rollout} \leftarrow g_I^{rollout} + g_J$ ;                 // $J$ corresponds to last expanded node
    **end**
    $g_I^{average} \leftarrow g_I^{average} + g_I^{rollout}$;
**end**
$g_I^{average} \leftarrow g_I^{average}/N$;
```

6.3.4 Propagation of the node cost

In this section, we let $\boldsymbol{G}_L^{aggr}$ be a variable that contains the total cost of the node S_L, where L could be any multi-index. According to the previous section, we let g_L be any of the following evaluation criteria g_L^{pcost}, g_L^{feef} and $g_L^{classic}$. Initially, $\boldsymbol{G}_L^{aggr}$ equals g_L. Also, we let S_K be the node that was selected for expansion, and let S_I be an arbitrary hidden state expanded from S_K. The cost of the newly expanded node(s) can be propagated either forward or backward. The forward propagation (towards the leaves) leads to the following equation:

$$\boldsymbol{G}_I^{aggr} \leftarrow g_I + \boldsymbol{G}_K^{aggr}, \tag{40}$$

where here $\boldsymbol{G}_K^{aggr}$ is the aggregated cost of the parent of S_I. Importantly, the symbol ← refers to a programming-like assignment (i.e., an incremental update) performed each time the tree is expanded. The backward propagation (towards the root) leads to:

$$\boldsymbol{G}_J^{aggr} \leftarrow \boldsymbol{G}_J^{aggr} + g_I \quad \forall J \in \mathbb{A}_I \tag{41}$$

where $\mathbb{A}_I$ corresponds to all ancestors of the newly expanded node S_I. We will see in Section 7 that these strategies respectively relate to active inference and sophisticated inference (Friston et al, 2021). Finally, since the agent is free to choose any action, we can back-propagate the (locally) minimum cost, i.e.,

$$\boldsymbol{G}_J^{aggr} \leftarrow \boldsymbol{G}_J^{aggr} + \min_{a \in \{1,\dots,|U|\}} g_{K::a} \quad \forall J \in \mathbb{A}_I, \tag{42}$$

where $K :: a$ is a multi-index obtained from K by adding the action a to the sequence of actions described by K. In all cases, the propagation step updates the counter n_J associated with each ancestor S_J of the newly expanded hidden state S_I; this counts the number of times the node S_J has been explored (exactly as in MCTS). This counter will be used for action selection, as well as for the computation of the average cost of S_J—$\bar{g}_J$—that was left undefined by Section 6.3.1. Formally, $\bar{g}_J$ is given by:

$$\bar{g}_J = \frac{1}{n_J} \boldsymbol{G}_J^{aggr}. \tag{43}$$

Remark 2 *The forward propagation of the cost presented above will only be used for theoretical purpose in Section 7. Practical implementation of BTAI should use the backward schemes.*

6.4 Action selection

The planning procedure presented in the previous section ends after a pre-specified amount of time has elapsed or when a sufficiently good policy has been found. When the planning is over, the agent needs to choose an action to act in its environment. In a companion paper (Champion et al, 2021a) that presents empirical results of BTAI, the actions are sampled from $\sigma(-\gamma \frac{g}{N})$, where $\sigma(\bullet)$ is a softmax function, γ is a precision parameter, g is a vector whose elements correspond to the cost of the root's children and N is a vector whose elements correspond to the number of visits of the root's children. Importantly, actions with low average cost are more likely to be selected than actions with high average cost.

Alternative approaches to action selection (Browne et al, 2012) could be studied. For example, one could imagine sampling actions from a categorical distribution with parameter $\sigma(N)$, where N is a vector containing the n_J of all children of the root node. Or, we could select the action corresponding to the root's child with the highest number of explorations n_J. The fact that it has been visited more often means that is has a lower cost overall. If there were a tie between several actions, the action with the lowest cost would be selected. The study of these strategies is left to future research.

6.5 Action-perception cycle with tree search

In active inference, the action-perception cycle realises an active inference agent in an infinite loop (van de Laar and de Vries, 2019). Each loop iteration begins with the agent sampling an observation from the environment. The observation is used to perform inference about the states and contingencies of the world, e.g., an impression on the retina might be used to reconstruct a three dimensional scene with a representation of the objects that it contains. Then, planning is performed by inferring the consequences of alternative action sequences. Importantly, only a subset of all possible action sequences are evaluated, due to the dynamical expansion of the generative model. Finally, the agent selects an action to perform in the environment by sampling a softmax function of minus the average cost weighted by the precision parameter γ, i.e., $\sigma(-\gamma\frac{g}{N})$. Therefore, actions with low average cost are more likely to be selected than actions with high average cost. We summarise our method using pseudo-code in Algorithm 2.

Algorithm 2: Action-perception cycle with tree search

```
while end of trial not reached do
    sample an observation from the environment;
    perform inference using the observation (Section 6.2);
    while maximum planning iteration not reached do
        select a node to be expanded (Section 6.3.1);
        perform the expansion of the node (Section 6.3.2);
        perform inference on the newly expanded nodes (Section 6.2);
        evaluate the cost of the newly expanded nodes (Section 6.3.3);
        propagate the cost of the nodes through the tree, either forward or backward (Section 6.3.4);
    end
    select an action to be performed (Section 6.4);
    execute the action in the environment leading to a new observation;
end
```

7. Connection between BTAI, active inference and sophisticated inference

In this section, we explore the relationship between BTAI, active inference (AcI) and sophisticated inference (SI). We show that BTAI is a class of algorithms that generalizes AcI and is related to SI. To do so, we focus on the "cost" of a policy for each method. In addition, we need to introduce the notion of localized and aggregated cost. The localized cost of a node S_I, denoted $\boldsymbol{G}_I^{local}$, is the cost of S_I in and of itself, i.e., without any consideration of the cost of past or future states. The aggregated cost of a node S_I, denoted $\boldsymbol{G}_I^{aggre}$, is the cost of S_I when taking into account either the cost of future states that can be reached from S_I (which is the case in SI) or the cost of the past states that an agent has to go through in order to reach S_I (which is the case in AcI).

7.1 Active inference

The full framework of active inference was described in Section 4. This section focuses on expressing the expected free energy in a recursive form that highlights the relationship between BTAI and AcI. We start by defining the notion of localized and aggregated EFE with Definitions 3 and 4, respectively. Then, we show that in active inference (under some assumptions described below), the aggregated EFE of a policy of size N is given by the aggregated EFE of a policy of size $N-1$ plus the localized EFE received at time $t+N$.

In active inference, a policy is a sequence of actions $\pi = (U_t, U_{t+1}, ..., U_{T-1})$, where T is the time horizon of planning, and for convenience, π_N denotes a policy of size N, obtained by selecting the first N actions of the policy π, i.e., $\pi_N = (U_t, U_{t+1}, ..., U_{t+N-1})$ with $N \leq T - t$. Recall from Section 4, that (in active inference) the expected free energy of a policy is given by:

$$\boldsymbol{G}(\pi) = \sum_{\tau=t+1}^{T} \boldsymbol{G}(\pi, \tau) = \sum_{\tau=t+1}^{T} \Bigg[D_{\mathrm{KL}}[Q(O_\tau|\pi)||P(O_\tau)] \; + \; \mathbb{E}_{Q(S_\tau|\pi)}[\mathrm{H}[P(O_\tau|S_\tau)]] \Bigg]. \tag{44}$$

If instead of letting τ range from $t+1$ to T, we let N range from 1 to $T-t$, then Equation 44 can be re-written as:

$$\boldsymbol{G}(\pi) = \sum_{N=1}^{T-t} \boldsymbol{G}(\pi, t+N) = \sum_{N=1}^{T-t} \Bigg[D_{\mathrm{KL}}[Q(O_{t+N}|\pi)||P(O_{t+N})] \; + \; \mathbb{E}_{Q(S_{t+N}|\pi)}[\mathrm{H}[P(O_{t+N}|S_{t+N})]] \Bigg]. \tag{45}$$

Additionally, under the assumption that the probability of observations and states are independent of future actions, i.e., that $\forall j \in \mathbb{N}_{>0}, Q(O_{t+i}|\pi_i) \approx Q(O_{t+i}|\pi_{i+j})$ and $\forall j \in \mathbb{N}_{>0}, Q(S_{t+i}|\pi_i) \approx Q(S_{t+i}|\pi_{i+j})$, π can be replaced by π_N in the RHS of the above equation, leading to:

$$\boldsymbol{G}(\pi) = \sum_{N=1}^{T-t} \boldsymbol{G}(\pi_N, t+N) = \sum_{N=1}^{T-t} \Bigg[D_{\mathrm{KL}}[Q(O_{t+N}|\pi_N)||P(O_{t+N})] \; + \; \mathbb{E}_{Q(S_{t+N}|\pi_N)}[\mathrm{H}[P(O_{t+N}|S_{t+N})]] \Bigg]. \tag{46}$$

Importantly, the elements of the above summation constitute the localized cost presented in Definition 3.

Definition 3 *We define the* ***localized cost*** *received at time $t+N$ after selecting policy π_N as:*

$$\boldsymbol{G}^{local}_{\pi_N} = \boldsymbol{G}(\pi_N, t+N) = D_{\mathrm{KL}}[Q(O_{t+N}|\pi_N)||P(O_{t+N})] \; + \; \mathbb{E}_{Q(S_{t+N}|\pi_N)}[H[P(O_{t+N}|S_{t+N})]]. \tag{47}$$

Importantly, the localized cost quantifies the amount of risk and ambiguity received by the agent at time step $t+N$, assuming that it will follow the policy π_N. We now turn to the notion of aggregated cost of a policy of size N. Definition 4 states that the aggregated cost of a policy is defined recursively. Indeed, by definition, a policy of size zero has an aggregated cost of zero, and then, the aggregated cost of a policy π_N (of size N) is equal to the the aggregated cost of π_{N-1} (of size $N-1$) plus the localized cost received at time $t+N$.

Definition 4 *We define the* ***aggregated cost*** *of a policy π_N of size N as:*

$$\boldsymbol{G}^{aggre}_{\pi_N} = \begin{cases} 0 & \textit{if } N = 0 \\ \boldsymbol{G}^{aggre}_{\pi_{N-1}} + \boldsymbol{G}^{local}_{\pi_N} & \textit{otherwise} \end{cases}. \tag{48}$$

Equipped with Definitions 3 and 4, we are now ready to state and prove Theorem 5 using the two Lemmas of Appendix E.

Theorem 5 *Under the assumption that the probability of observations and states are independent of future actions, i.e., $\forall j \in \mathbb{N}_{>0}, Q(O_{t+i}|\pi_i) \approx Q(O_{t+i}|\pi_{i+j})$ and $\forall j \in \mathbb{N}_{>0}, Q(S_{t+i}|\pi_i) \approx Q(S_{t+i}|\pi_{i+j})$, the expected free energy can be written as:*

$$\boldsymbol{G}(\pi_N) \approx \boldsymbol{G}^{aggre}_{\pi_N} = \boldsymbol{G}^{aggre}_{\pi_{N-1}} + \boldsymbol{G}^{local}_{\pi_N}. \tag{49}$$

Proof This proof is based on two lemmas demonstrated in Appendix E. Note that in active inference the expected free energy is defined as:

$$\boldsymbol{G}(\pi) = \sum_{\tau=t+1}^{T} \boldsymbol{G}(\pi, \tau). \tag{50}$$

Let N denote the size of the policy π, i.e. $N = T - t$. Note that because π is of size N, then by definition $\pi = \pi_N$, and the above equation can be re-written as:

$$\boldsymbol{G}(\pi) = \boldsymbol{G}(\pi_N) = \sum_{\tau=t+1}^{t+N} \boldsymbol{G}(\pi_N, \tau). \tag{51}$$

Expanding the summation and using Definition 3:

$$\begin{aligned} \boldsymbol{G}(\pi_N) &= \sum_{\tau=t+1}^{t+N-1} \boldsymbol{G}(\pi_N, \tau) + \boldsymbol{G}(\pi_N, t+N) & (52) \\ &= \sum_{\tau=t+1}^{t+N-1} \boldsymbol{G}(\pi_N, \tau) + \boldsymbol{G}^{local}_{\pi_N}. & (53) \end{aligned}$$

If, instead of letting τ range from $t+1$ to $t+N-1$, we let i range from 1 to $N-1$, then the above equation can be re-written as:

$$\boldsymbol{G}(\pi_N) = \sum_{i=1}^{N-1} \boldsymbol{G}(\pi_N, t+i) + \boldsymbol{G}^{local}_{\pi_N}. \tag{54}$$

Note that $\forall i \in \{1, ..., N-1\}, N > i$, and thus there exists a $k_i \in \mathbb{N}_{>0}$ such that $\forall i \in \{1, ..., N-1\}, N = i + k_i$. Therefore, we replace N by $i + k_i$ in the above summation:

$$\boldsymbol{G}(\pi_N) = \sum_{i=1}^{N-1} \boldsymbol{G}(\pi_{i+k_i}, t+i) + \boldsymbol{G}_{\pi_N}^{local}. \tag{55}$$

Lemma 11 tells us that under the assumption that the probability of observations and states are independent of future actions, $\forall k_i \in \mathbb{N}_{>0}, \boldsymbol{G}(\pi_{i+k_i}, t+i) \approx \boldsymbol{G}(\pi_i, t+i)$, which allows us to remove the k_i to get:

$$\boldsymbol{G}(\pi_N) \approx \sum_{i=1}^{N-1} \boldsymbol{G}(\pi_i, t+i) + \boldsymbol{G}_{\pi_N}^{local}. \tag{56}$$

Finally, Lemma 12 states that $\sum_{i=1}^{N-1} \boldsymbol{G}(\pi_i, t+i) = \boldsymbol{G}_{\pi_{N-1}}^{aggre}$, and thus:

$$\boldsymbol{G}(\pi_N) \approx \boldsymbol{G}_{\pi_{N-1}}^{aggre} + \boldsymbol{G}_{\pi_N}^{local} \triangleq \boldsymbol{G}_{\pi_N}^{aggre}. \tag{57}$$

The above equation will be used in Section 7.4 to show that BTAI generalizes active inference. ■

7.2 Sophisticated inference

Sophisticated inference (Friston et al, 2021) is a new type of active inference that defines the EFE recursively from the time horizon backward. Intuitively, the agent does not simply ask "what would happen if I did that", but instead wonders "what would I believe about what would happen if I did that". In other words, the agent is exhibiting a form of sophistication, which refers to the fact of having beliefs about one's own or another's beliefs. Friston et al (2021) also replaced variational message passing by an alternative inference scheme called Bayesian Filtering (Fox et al, 2003). While the change of inference method is of little relevance to us here, the recursive definition of the EFE is at the core of this section. As explained in Section 4.3 of Da Costa et al (2020b), the (recursive) EFE of a Markov decision process is given by:

$$G(U_{T-1}, S_{T-1}) = D_{\mathrm{KL}} \left[\, Q(S_T | U_{T-1}, S_{T-1}) || \, V(S_T) \right] \tag{58}$$

$$G(U_\tau, S_\tau) = D_{\mathrm{KL}} \left[\, Q(S_{\tau+1} | U_\tau, S_\tau) || \, V(S_{\tau+1}) \right] + \mathbb{E}_{Q(U_{\tau+1}, S_{\tau+1} | U_\tau, S_\tau)} [G(U_{\tau+1}, S_{\tau+1})] \tag{59}$$

where U_τ and S_τ are the action and state at time τ, and $V(S_\tau)$ is the target (i.e., desired) distribution over states at time τ. Using our terminology of localized and aggregated cost, this can be rewritten as:

$$\underbrace{G(U_{T-1}, S_{T-1})}_{\boldsymbol{G}^{aggre}(U_{T-1}, S_{T-1})} = \underbrace{D_{\mathrm{KL}}\left[\, Q(S_T|U_{T-1}, S_{T-1}) || \, V(S_T)\right]}_{\boldsymbol{G}^{local}(U_{T-1}, S_{T-1})} \tag{60}$$

$$\underbrace{G(U_\tau, S_\tau)}_{\boldsymbol{G}^{aggre}(U_\tau, S_\tau)} = \underbrace{D_{\mathrm{KL}}\left[\, Q(S_{\tau+1}|U_\tau, S_\tau) || \, V(S_{\tau+1})\right]}_{\boldsymbol{G}^{local}(U_\tau, S_\tau)} + \mathbb{E}_{Q(U_{\tau+1}, S_{\tau+1}|U_\tau, S_\tau)}\left[\, \underbrace{G(U_{\tau+1}, S_{\tau+1})}_{\boldsymbol{G}^{aggre}(U_{\tau+1}, S_{\tau+1})}\,\right] \tag{61}$$

Put simply, the aggregated cost of taking action U_τ in state S_τ can be computed by summing the localized cost at time step τ and the expected aggregated cost at time step $\tau + 1$, i.e.,

$$\boldsymbol{G}^{aggre}(U_\tau, S_\tau) = \boldsymbol{G}^{local}(U_\tau, S_\tau) + \mathbb{E}_{Q(U_{\tau+1}, S_{\tau+1}|U_\tau, S_\tau)}[\boldsymbol{G}^{aggre}(U_{\tau+1}, S_{\tau+1})] \tag{62}$$

Note that for $\tau = T - 1$ the second term vanishes because future states beyond the temporal horizon are ignored, and thus $\boldsymbol{G}^{aggre}(U_{T-1}, S_{T-1}) = \boldsymbol{G}^{local}(U_{T-1}, S_{T-1})$. Also, the above equation will be useful in Section 7.5 to show that BTAI is related to sophisticated inference.

Remark 6 *The recursive aspect of Equation 59 is deeply related to dynamic programming and the interested reader is referred to Da Costa et al (2020b) for details about this relationship.*

7.3 Branching Time Active Inference (BTAI)

In BTAI, the (localized) cost of the hidden state S_I is defined as $\boldsymbol{G}^{local}_I = g_I$, where g_I can be equal to $g^{classic}_I$, g^{feef}_I or g^{pcost}_I, and there are two ways of computing the aggregated cost of S_I. We can either propagate the localized cost towards the leaves (forward):

$$g_I \leftarrow g_I + g_{I \backslash last} \tag{63}$$

where the $g_{I \backslash last}$ is the cost of the parent of S_I. Alternatively, we can back-propagate the cost towards the root

$$g_J \leftarrow g_J + g_I \quad \forall J \in \mathbb{A}, \tag{64}$$

where $\mathbb{A}$ corresponds to all ancestors of the newly expanded node S_I.

7.4 BTAI as a generalisation of active inference

To understand the relationship between BTAI and active inference, we need to focus on the forward propagation of the cost where the cost is given by $g_I^{classic}$. Recall that the update for forward propagation is given by:

$$g_I^{classic} \leftarrow g_I^{classic} + g_{I \backslash last}^{classic}, \tag{65}$$

where the $g_{I \backslash last}$ is the cost of the parent of S_I, i.e., the parent of the newly expanded node. This equation tells us that the aggregated cost of S_I is equal to the localized cost of S_I plus the aggregated cost of $S_{I \backslash last}$, i.e.,

$$\underbrace{g_I^{classic}}_{\boldsymbol{G}_I^{aggre}} \leftarrow \underbrace{g_I^{classic}}_{\boldsymbol{G}_I^{local}} + \underbrace{g_{I \backslash last}^{classic}}_{\boldsymbol{G}_{I \backslash last}^{aggre}} \quad \Leftrightarrow \quad \boldsymbol{G}_I^{aggre} = \boldsymbol{G}_I^{local} + \boldsymbol{G}_{I \backslash last}^{aggre}, \tag{66}$$

but, then, we also recall (49), i.e.,

$$\boldsymbol{G}_{\pi_N}^{aggre} \approx \boldsymbol{G}_{\pi_{N-1}}^{aggre} + \boldsymbol{G}_{\pi_N}^{local} \quad \Leftrightarrow \quad \boldsymbol{G}_{\pi_N}^{aggre} \approx \boldsymbol{G}_{\pi_N}^{local} + \boldsymbol{G}_{\pi_{N-1}}^{aggre}. \tag{67}$$

The only difference between Equations 49 and 66 is notational. Indeed, in BTAI (Eq. 49) a policy is represented by a multi-index denoting the sequence of actions selected, e.g., $I = (1, 2)$ corresponds to a policy of size two consisting of action one followed by action two. In contrast, in active inference, a policy is a sequence of actions, e.g., $\pi_2 = (1, 2)$ corresponds to the same policy as the one described by I.

7.5 Relationship between BTAI and sophisticated inference

The relationship between BTAI and sophisticated inference is slightly more involved. The backward propagation equation, i.e.,

$$g_J \leftarrow g_J + g_I \quad \forall J \in \mathbb{A}, \tag{68}$$

tells us that when expanding a node S_I, we first need to compute its localized cost g_I and then add g_I to the aggregated cost of its ancestors S_J where $J \in \mathbb{A}$. In other words, we can rewrite the backward propagation equation as the following: the aggregated cost of an arbitrary node S_J will equal the sum of its localized cost g_J and that of its descendants $\mathbb{D}_J$ that have already been evaluated

$$\boldsymbol{G}_J^{aggre} = \boldsymbol{G}_J^{local} + \sum_{S_K \in \mathbb{D}_J} \boldsymbol{G}_K^{local}, \tag{69}$$

where the descendants are the children, children of children, etc. We can further simplify this expression by grouping the summands by children of S_J. This leads us to:

$$\boldsymbol{G}_J^{aggre} = \boldsymbol{G}_J^{local} + \sum_{S_I \in \mathrm{ch}_J} \underbrace{\left(\boldsymbol{G}_I^{local} + \sum_{S_K \in \mathbb{D}_I} \boldsymbol{G}_K^{local} \right)}_{\boldsymbol{G}_I^{aggre}} \tag{70}$$

$$= \boldsymbol{G}_J^{local} + \sum_{S_I \in \mathrm{ch}_J} \boldsymbol{G}_I^{aggre}, \tag{71}$$

where ch_J are the children of S_J and $\mathbb{D}_I$ are the descendants of S_I. The above equation has clear similarities to Equation (62), which is,

$$\boldsymbol{G}^{aggre}(U_\tau, S_\tau) = \boldsymbol{G}^{local}(U_\tau, S_\tau) + \mathbb{E}_{Q(U_{\tau+1}, S_{\tau+1}|U_\tau, S_\tau)}[\boldsymbol{G}^{aggre}(U_{\tau+1}, S_{\tau+1})]. \tag{72}$$

However, the second term of the RHS of (62) is an expectation, while the second term of the RHS of (70) is a summation over the children that have already been expanded. The expectation in (62) is w.r.t.

$$Q(U_{\tau+1}, S_{\tau+1}|U_\tau, S_\tau) = Q(U_{\tau+1}|S_{\tau+1})Q(S_{\tau+1}|U_\tau, S_\tau), \tag{73}$$

where:

$$Q(U_{\tau+1}|S_{\tau+1}) > 0 \Leftrightarrow U_{\tau+1} \in \operatorname*{arg\,mins}_{U \in \mathbb{U}} \boldsymbol{G}^{aggre}(U, S_{\tau+1}), \tag{74}$$

where $\mathbb{U}$ is the set of all possible actions, and argmins is defined as:

$$M = \operatorname*{arg\,mins}_{U \in \mathbb{U}} f(U) \Leftrightarrow M = \Big\{ U \in \mathbb{U} \;\Big|\; f(U) = \min_{U' \in \mathbb{U}} f(U') \Big\}. \tag{75}$$

This means that an action is assigned positive probability mass if and only if it minimises the aggregated cost at the next time point $\boldsymbol{G}^{aggre}(U_{\tau+1}, S_{\tau+1})$ and a set is required, because multiple actions could have the same minimum cost. Note that if there is a unique minimum, then $Q(U_{\tau+1}|S_{\tau+1})$ will be a one-hot like distribution with a probability of one for the best action.

To conclude, Equations (62) and (70) suggest that BTAI and SI share a similar notion of EFE, where the immediate (or localized) EFE is added to the future (or aggregated) EFE. Both BTAI and SI propagate the cost backward, however, in SI the aggregated EFE (i.e., the back-propagated cost) is weighted by the probability of the next action and states, i.e., $Q(U_{\tau+1}, S_{\tau+1}|U_\tau, S_\tau)$. Intuitively, the weighting terms in SI discounts the impact of the back-propagated cost for unlikely states and (locally) sub-optimal actions. Importantly, those weighting terms emerge from the recursive definition of the EFE that relates to the Bellman equation (Da Costa et al, 2020b). In contrast, there are no such weights in BTAI because BTAI finds its inspiration in active inference.

8. Conclusion and future works

In this paper, we have presented a new approach where planning is cast as structure learning. Simply put, this approach consists of dynamically expanding branches of the generative model by evaluating alternative futures under different action sequences. The dynamic expansion trades off evaluating promising (with repect to the target distribution) policies with exploring policies whose outcomes are uncertain. We proposed two different tree search methods: the first in which the nodes' cost is propagated forward from the root node to the leaves; the second in which the nodes' cost is propagated backward from the leaves to the root node. Then, in Section 7 we showed that forward propagation of the EFE leads to active inference (AcI) under the assumption that the probability of observations and states are independent of future actions, and that backward propagation relates to sophisticated inference (SI). This clarifies the link between AcI and BTAI, and helps to understand the relationship between AcI and SI.

Importantly, by performing a complexity class analysis, we have shown that while Active Inference suffers from an exponential complexity class, our approach scales nicely (linearly) with the number of tree expansions, c.f., Section 3.4.2 of our companion paper (Champion et al, 2021a). Of course, the total number of possible expansions grows exponentially but as has been empirically shown in the reinforcement learning literature, even complex tasks, such as chess or the game of go, can be performed efficiently with MCTS (Silver et al, 2016; Schrittwieser et al, 2019), i.e., the efficiency of MCTS-like approaches relies on the ability to guide the expansion procedure using either powerful heuristics (like the EFE) or neural networks or both.

We also know that humans engage in counterfactual reasoning (Rafetseder et al, 2013), which, in our planning context, could involve the consideration and evaluation of alternative (non-selected) sequences of decisions. It may be that, because of the more exhaustive representation of possible trajectories, the classic active inference can more efficiently engage in counterfactual reasoning. In contrast, branching-time active inference would require these alternatives to be generated “a fresh” for each counterfactual deliberation. In this sense, one might argue that there is a trade-off: branching-time active inference provides considerably more efficient planning to attain current goals, classic active inference provides a more exhaustive assessment of paths not taken.

Now that we have laid out the mathematics of BTAI, many directions of research could be investigated. One could for example obtain an intuitive understanding of the model's parameters through experimental study. At first it might be necessary to restrict oneself to agents without learning, i.e., inference only. This step should help answer questions such as: How does the number of expansions of the tree and the quality of the prior preferences impact the quality of planning? What is the best inference method (i.e., local or global inference) to use during planning?

Then, one could consider learning of the transition and likelihood matrices as well as the vector of initial states. This can be done in at least two ways. The first is to add Dirichlet priors over those matrices/vectors and the second would be to use neural networks as function approximators. The second option will lead to a deep active inference agent (Sancaktar and Lanillos, 2020; Millidge, 2020) equipped with tree search that could be directly compared to the method of Fountas et al (2020). Including deep neural networks in the framework will also enable direct comparison with the deep reinforcement learning literature (Haarnoja et al, 2018; Mnih et al, 2013; van Hasselt et al, 2016; Lample and Chaplot, 2017; Silver et al, 2016). These comparisons will enable the impact of epistemic terms to be studied when the agent is composed of deep neural networks.

Another, very important direction for future research would be the creation of a biologically plausible implementation of BTAI. For example, using artificial neural networks to model the various mappings of the framework may provide a neural-based implementation of BTAI that is closer to biology. This would especially be the case, if the back-propagation algorithm frequently used for learning is replaced by the (more biologically plausible) generalized recirculation algorithm (O'Reilly, 1996). Another possible approach would be to use populations of neurons to encode the update equations of the framework, as was proposed by Friston et al (2017).

Whatever technique is chosen for learning and inference, implementing MCTS in a biologically plausible way will be challenging. Indeed, MCTS requires a dynamic expansion of the search tree used to explore the space of possible policies. Each time an expansion is performed, the agent needs to store the associated variables such as: the number of visits, the aggregated expected free energy, and the posterior beliefs of the newly expanded node. Given the fast pace at which planning must be performed to be useful, slow mechanisms such as synaptic plasticity and neurogenesis are likely to be unsuitable for the task. A more plausible approach might rely upon a change of neuronal activation, which can occur within a few hundred milliseconds. One such approach uses a binding pool (Bowman and Wyble, 2007) and provides a notion of variable. In this framework, a variable is composed of two parts. First, a *token* that can be intuitively understood as the variable's name, and second, a *type* corresponding to the variable's value. The binding pool is then composed of neurons representing the fact that a variable's name is *bound* (or set) to a specific value. A localist realisation of a binding pool could be implemented as a 2D array of neurons of size "number of tokens" × "number of types". However, such a representation is quite inefficient and a more compact (i.e. distributed) representation has been developed (Wyble and Bowman, 2006). If variables are complex data structures, such as those required by BTAI, they can be realised in a neural substrate.

Finally, in this paper, we focused on the UCT criterion for node selection because it is a standard choice in the reinforcement learning literature (Silver et al, 2016; Schrittwieser et al, 2019). However, it would be valuable to consider alternative criteria such as Thompson Sampling (Thompson, 1933; Russo et al, 2018; Sajid et al,

2021) or expected improvement (Brochu et al, 2010; Bergstra et al, 2011). For example, Thompson Sampling has been shown to improve upon the standard UCT criterion when applied to MCTS (Bai et al, 2013), but requires additional modelling and we leave this for future research.

Acknowledgments

We would like to thank the reviewers for their valuable feedback, which greatly improved the quality of the present paper. LD is supported by the Fonds National de la Recherche, Luxembourg (Project code: 13568875). This publication is based on work partially supported by the EPSRC Centre for Doctoral Training in Mathematics of Random Systems: Analysis, Modelling and Simulation (EP/S023925/1).

References

Auer P, Cesa-Bianchi N, Fischer P (2002) Finite-time analysis of the multiarmed bandit problem. Machine Learning 47(2):235–256, DOI 10.1023/A:1013689704352, URL https://doi.org/10.1023/A:1013689704352

Bai A, Wu F, Chen X (2013) Bayesian mixture modelling and inference based Thompson sampling in Monte-Carlo tree search. In: Proceedings of the Advances in Neural Information Processing Systems (NIPS), Lake Tahoe, United States, pp 1646–1654

Bergstra J, Bardenet R, Bengio Y, Kégl B (2011) Algorithms for hyper-parameter optimization. In: Shawe-Taylor J, Zemel R, Bartlett P, Pereira F, Weinberger KQ (eds) Advances in Neural Information Processing Systems, Curran Associates, Inc., vol 24, URL https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf

Botvinick M, Toussaint M (2012) Planning as inference. Trends in Cognitive Sciences 16(10):485 – 488, DOI https://doi.org/10.1016/j.tics.2012.08.006

Bowman H (2005) Concurrency Theory: Calculi an Automata for Modelling Untimed and Timed Concurrent Systems. Springer, Dordrecht, URL https://cds.cern.ch/record/1250124

Bowman H, Wyble B (2007) The simultaneous type, serial token model of temporal attention and working memory. Psychological Review 114(1):38–70

Brochu E, Cora VM, de Freitas N (2010) A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. arXiv preprint arXiv:10122599

Browne CB, Powley E, Whitehouse D, Lucas SM, Cowling PI, Rohlfshagen P, Tavener S, Perez D, Samothrakis S, Colton S (2012) A survey of Monte Carlo tree search methods. IEEE Transactions on Computational Intelligence and AI in Games 4(1):1–43

Çatal O, Verbelen T, Nauta J, Boom CD, Dhoedt B (2020) Learning perception and planning with deep active inference. In: 2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020, IEEE, pp 3952–3956, DOI 10.1109/ICASSP40776.2020.9054364, URL https://doi.org/10.1109/ICASSP40776.2020.9054364

Champion T, Bowman H, Grześ M (2021a) Branching time active inference: empirical study. URL https://arxiv.org/abs/2111.11276, available at https://arxiv.org/abs/2111.11276.

Champion T, Grześ M, Bowman H (2021b) Realizing Active Inference in Variational Message Passing: The Outcome-Blind Certainty Seeker. Neural Computation pp 1–65, DOI 10.1162/neco_a_01422, URL https://doi.org/10.1162/neco_a_01422, https://direct.mit.edu/neco/article-pdf/doi/10.1162/neco_a_01422/1930278/neco_a_01422.pdf

Cox M, van de Laar T, de Vries B (2019) A factor graph approach to automated design of Bayesian signal processing algorithms. Int J Approx Reason 104:185–204, DOI 10.1016/j.ijar.2018.11.002, URL https://doi.org/10.1016/j.ijar.2018.11.002

Cullen M, Davey B, Friston KJ, Moran RJ (2018) Active inference in openai gym: A paradigm for computational investigations into psychiatric illness. Biological Psychiatry: Cognitive Neuroscience and Neuroimaging 3(9):809 – 818, DOI https://doi.org/10.1016/j.bpsc.2018.06.010, URL http://www.sciencedirect.com/science/article/pii/S2451902218301617, computational Methods and Modeling in Psychiatry

Da Costa L, Parr T, Sajid N, Veselic S, Neacsu V, Friston K (2020a) Active inference on discrete state-spaces: A synthesis. Journal of Mathematical Psychology 99:102,447, DOI https://doi.org/10.1016/j.jmp.2020.102447, URL https://www.sciencedirect.com/science/article/pii/S0022249620300857

Da Costa L, Sajid N, Parr T, Friston K, Smith R (2020b) The relationship between dynamic programming and active inference: the discrete, finite-horizon case. arXiv 2009.08111

FitzGerald THB, Dolan RJ, Friston K (2015) Dopamine, reward learning, and active inference. Frontiers in Computational Neuroscience 9:136, DOI 10.3389/fncom.2015.00136, URL https://www.frontiersin.org/article/10.3389/fncom.2015.00136

Forney GD (2001) Codes on graphs: normal realizations. IEEE Transactions on Information Theory 47(2):520–548

Fountas Z, Sajid N, Mediano PAM, Friston KJ (2020) Deep active inference agents using Monte-Carlo methods. In: Larochelle H, Ranzato M, Hadsell R, Balcan M, Lin H (eds) Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, URL https://proceedings.neurips.cc/paper/2020/hash/865dfbde8a344b44095495f3591f7407-Abstract.html

Fox CW, Roberts SJ (2012) A tutorial on variational Bayesian inference. Artificial Intelligence Review 38(2):85–95, DOI 10.1007/s10462-011-9236-8, URL https://doi.org/10.1007/s10462-011-9236-8

Fox V, Hightower J, Liao L, Schulz D, Borriello G (2003) Bayesian filtering for location estimation. IEEE Pervasive Computing 2(3):24–33, DOI 10.1109/MPRV.2003.1228524

Friston K, Schwartenbeck P, Fitzgerald T, Moutoussis M, Behrens T, Dolan R (2013) The anatomy of choice: active inference and agency. Frontiers in Human Neuroscience 7:598, DOI 10.3389/fnhum.2013.00598, URL https://www.frontiersin.org/article/10.3389/fnhum.2013.00598

Friston K, FitzGerald T, Rigoli F, Schwartenbeck P, Doherty JO, Pezzulo G (2016) Active inference and learning. Neuroscience & Biobehavioral Reviews 68:862 – 879, DOI https://doi.org/10.1016/j.neubiorev.2016.06.022

Friston K, Da Costa L, Hafner D, Hesp C, Parr T (2021) Sophisticated Inference. Neural Computation 33(3):713–763, DOI 10.1162/neco_a_01351, URL https://doi.org/10.1162/neco_a_01351, https://direct.mit.edu/neco/article-pdf/33/3/713/1889421/neco_a_01351.pdf

Friston KJ, Parr T, de Vries B (2017) The graphical brain: Belief propagation and active inference. Network Neuroscience 1(4):381–414, DOI 10.1162/NETN_a_00018, URL https://doi.org/10.1162/NETN_a_00018, https://doi.org/10.1162/NETN_a_00018

van Glabbeek RJ (1993) The linear time — branching time spectrum II. In: Best E (ed) CONCUR'93, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 66–81

Glabbeek RJv (1990) The linear time-branching time spectrum (extended abstract). In: Proceedings of the Theories of Concurrency: Unification and Extension, Springer-Verlag, Berlin, Heidelberg, CONCUR '90, p 278–297

Haarnoja T, Zhou A, Abbeel P, Levine S (2018) Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. CoRR abs/1801.01290, URL http://arxiv.org/abs/1801.01290, 1801.01290

Hafner D, Ortega PA, Ba J, Parr T, Friston KJ, Heess N (2020) Action and perception as divergence minimization. CoRR abs/2009.01791, URL https://arxiv.org/abs/2009.01791, 2009.01791 van Hasselt H, Guez A, Silver D (2016) Deep reinforcement learning with double q-learning. In: Schuurmans D, Wellman MP (eds) Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA, AAAI Press, pp 2094–2100, URL http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12389

Itti L, Baldi P (2009) Bayesian surprise attracts human attention. Vision Research 49(10):1295 – 1306, DOI https://doi.org/10.1016/j.visres.2008.09.007, URL http://www.sciencedirect.com/science/article/pii/S0042698908004380, visual Attention: Psychophysics, electrophysiology and neuroimaging

Kirchhoff M, Parr T, Palacios E, Friston K, Kiverstein J (2018) The markov blankets of life: autonomy, active inference and the free energy principle. Journal of The Royal Society Interface 15(138):20170,792, DOI 10.1098/rsif.2017.0792, URL https://royalsocietypublishing.org/doi/abs/10.1098/rsif.2017.0792, https://royalsocietypublishing.org/doi/pdf/10.1098/rsif.2017.0792

Kocsis L, Szepesvári C (2006) Bandit based Monte-Carlo planning. In: Fürnkranz J, Scheffer T, Spiliopoulou M (eds) Machine Learning: ECML 2006, 17th European Conference on Machine Learning, Berlin, Germany, September 18-22, 2006, Proceedings, Springer, Lecture Notes in Computer Science, vol 4212, pp 282–293, DOI 10.1007/11871842_29, URL https://doi.org/10.1007/11871842_29

Kocsis L, Szepesvári C (2006) Bandit based Monte-Carlo planning. In: Fürnkranz J, Scheffer T, Spiliopoulou M (eds) Machine Learning: ECML 2006, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 282–293

van de Laar T, de Vries B (2019) Simulating active inference processes by message passing. Front Robotics and AI 2019, DOI 10.3389/frobt.2019.00020, URL https://doi.org/10.3389/frobt.2019.00020

Lai T, Robbins H (1985) Asymptotically efficient adaptive allocation rules. Adv Appl Math 6(1):4–22, DOI 10.1016/0196-8858(85)90002-8, URL https://doi.org/10.1016/0196-8858(85)90002-8

Lample G, Chaplot DS (2017) Playing FPS games with deep reinforcement learning. In: Singh SP, Markovitch S (eds) Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA, AAAI Press, pp 2140–2146, URL http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14456

Maisto D, Gregoretti F, Friston KJ, Pezzulo G (2021) Active tree search in large POMDPs. CoRR abs/2103.13860, URL https://arxiv.org/abs/2103.13860, 2103.13860

Millidge B (2019) Combining active inference and hierarchical predictive coding: A tutorial introduction and case study. URL https://doi.org/10.31234/osf.io/kf6wc

Millidge B (2020) Deep active inference as variational policy gradients. Journal of Mathematical Psychology 96:102,348, DOI https://doi.org/10.1016/j.jmp.2020.102348, URL http://www.sciencedirect.com/science/article/pii/S0022249620300298

Millidge B, Tschantz A, Buckley CL (2021) Whence the Expected Free Energy? Neural Computation 33(2):447–482, DOI 10.1162/neco_a_01354, URL https://doi.org/10.1162/neco_a_01354, https://direct.mit.edu/neco/article-pdf/33/2/447/1896836/neco_a_01354.pdf

Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller MA (2013) Playing atari with deep reinforcement learning. CoRR abs/1312.5602, URL http://arxiv.org/abs/1312.5602, 1312.5602

O'Reilly RC (1996) Biologically plausible error-driven learning using local activation differences: The generalized recirculation algorithm. Neural Comput 8(5):895–938, DOI 10.1162/neco.1996.8.5.895, URL https://doi.org/10.1162/neco.1996.8.5.895

Parr T, Friston KJ (2019) Generalised free energy and active inference. Biological Cybernetics 113(5):495–513, DOI 10.1007/s00422-019-00805-w, URL https://doi.org/10.1007/s00422-019-00805-w

Pezzato C, Corbato CH, Wisse M (2020) Active inference and behavior trees for reactive action planning and execution in robotics. CoRR abs/2011.09756, URL https://arxiv.org/abs/2011.09756, 2011.09756

Rafetseder E, Schwitalla M, Perner J (2013) Counterfactual reasoning: From childhood to adulthood. Journal of experimental child psychology 114(3):389–404

Rosin CD (2010) Multi-armed bandits with episode context. In: International Symposium on Artificial Intelligence and Mathematics, ISAIM 2010, Fort Lauderdale, Florida, USA, January 6-8, 2010, URL http://gauss.ececs.uc.edu/Workshops/isaim2010/papers/rosin.pdf

Russo DJ, Van Roy B, Kazerouni A, Osband I, Wen Z (2018) A tutorial on Thompson sampling. Found Trends Mach Learn 11(1):1–96, DOI 10.1561/2200000070, URL https://doi.org/10.1561/2200000070

Sajid N, Ball PJ, Parr T, Friston KJ (2021) Active Inference: Demystified and Compared. Neural Computation 33(3):674–712, DOI 10.1162/neco_a_01357, URL https://doi.org/10.1162/neco_a_01357, https://direct.mit.edu/neco/article-pdf/33/3/674/1889396/neco_a_01357.pdf

Sancaktar C, Lanillos P (2020) End-to-end pixel-based deep active inference for body perception and action. ArXiv abs/2001.05847

Sancaktar C, van Gerven MAJ, Lanillos P (2020) End-to-end pixel-based deep active inference for body perception and action. In: Joint IEEE 10th International Conference on Development and Learning and Epigenetic Robotics, ICDL-EpiRob 2020, Valparaiso, Chile, October 26-30, 2020, IEEE, pp 1–8, DOI 10.1109/ICDL-EpiRob48136.2020.9278105, URL https://doi.org/10.1109/ICDL-EpiRob48136.2020.9278105

Schrittwieser J, Antonoglou I, Hubert T, Simonyan K, Sifre L, Schmitt S, Guez A, Lockhart E, Hassabis D, Graepel T, Lillicrap TP, Silver D (2019) Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model. ArXiv abs/1911.08265

Schwartenbeck P, Passecker J, Hauser TU, FitzGerald THB, Kronbichler M, Friston K (2018) Computational mechanisms of curiosity and goal-directed exploration. bioRxiv DOI 10.1101/411272, URL https://www.biorxiv.org/content/early/2018/09/07/411272, https://www.biorxiv.org/content/early/2018/09/07/411272.full.pdf

Silver D, Huang A, Maddison CJ, Guez A, Sifre L, van den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M, Dieleman S, Grewe D, Nham J, Kalchbrenner N, Sutskever I, Lillicrap TP, Leach M, Kavukcuoglu K, Graepel T, Hassabis D (2016) Mastering the game of Go with deep neural networks and tree search. Nature 529(7587):484–489, DOI 10.1038/nature16961, URL https://doi.org/10.1038/nature16961

Smith R, Friston KJ, Whyte CJ (2021) A step-by-step tutorial on active inference and its application to empirical data. URL https://psyarxiv.com/b4jm6/

Sondik EJ (1971) The optimal control of partially observable markov processes. PhD thesis, Stanford University URL https://ci.nii.ac.jp/naid/20000916958/en/

Thompson WR (1933) On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. Biometrika 25(3/4):285–294, URL http://www.jstor.org/stable/2332286

Winn J, Bishop C (2005) Variational message passing. Journal of Machine Learning Research 6:661–694

Wyble B, Bowman H (2006) A neural network account of binding discrete items into working memory using a distributed pool of flexible resources. Journal of Vision 6(6):33–33a

Appendix A: Generalized inner and outer products

Generalized outer products: Given N vectors V^i, the generalized outer product returns an N dimensional array W, whose element in position $(x_1, ..., x_N)$ is given by $V^1_{x_1} \times ... \times V^N_{x_N}$, where $V^i_{x_j}$ is the x_j-th element of the i-th vector. In other words:

$$W = \otimes\Big[V^1, ..., V^N\Big] \Leftrightarrow W(x_1, ..., x_N) = V^1_{x_1} \times ... \times V^N_{x_N} \quad \forall x_j \in \{1, ..., |V^j|\}\ \forall j \in \{1, ..., N\}, \tag{76}$$

where $|V^j|$ is the number of elements in V^j. Also, note that by definition W is a N-tensor of size $|V^1| \times ... \times |V^N|$. Figure 7 illustrates the generalized outer product for $N = 3$.

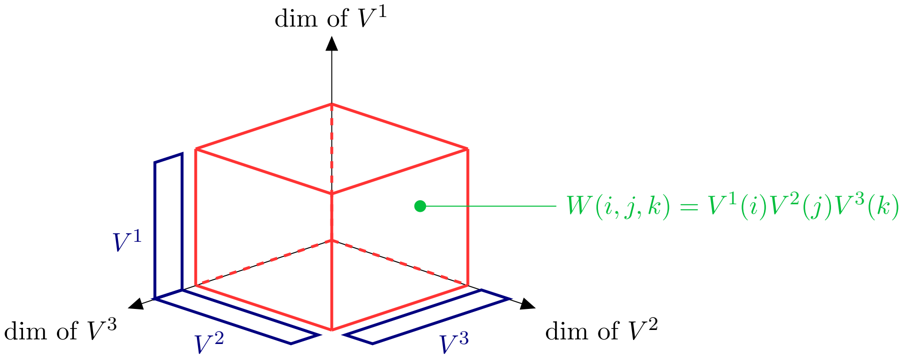


Figure 7: This figure illustrates the generalized outer product $W = \otimes\big[V^1, V^2, V^3\big]$, where W is a cube of values illustrated in red, whose typical element $W(i, j, k)$ is the product of $V^1(i)$, $V^2(j)$ and $V^3(k)$. Also, the vectors $V^i\ \forall i \in \{1, ..., 3\}$ are drawn in blue along the dimension of the cube they correspond to.

Generalized inner products: Given an N-tensor W and $M = N - 1$ vectors V^i, the generalized inner product returns a vector Z obtained by performing a weighted average (with weighting coming from the vectors) over all but one dimension. In other words:

$$Z = W \odot \Big[V^1, ..., V^M\Big] \Leftrightarrow Z(x_j) = \sum_{\substack{x_1 \in \{1,...,|V^1|\} \\ ... \\ x_M \in \{1,...,|V^M|\}}} V^1_{x_1} \times ... \times W(x_1, ..., x_j, ..., x_M) \times ... \times V^M_{x_M}$$

$$\forall x_j \in \{1, ..., |Z|\}, \tag{77}$$

where $|Z|$ denotes the number of elements in Z, and the large summand is over all x_r for $r \in \{1, ..., M\} \setminus \{j\}$, i.e., excluding j. Also, note that if $|W|_{V^i}\ \forall i \in \{1, ..., M\}$ is the number of elements in the dimension corresponding to V^i, then for $W \odot \big[V^1, ..., V^M\big]$ to be properly defined, we must have $|W|_{V^i} = |V^i|\ \forall i \in \{1, ..., M\}$ where $|V^i|$ is the number of elements in V^i. Figure 8 illustrates the generalized inner product for $N = 3$.

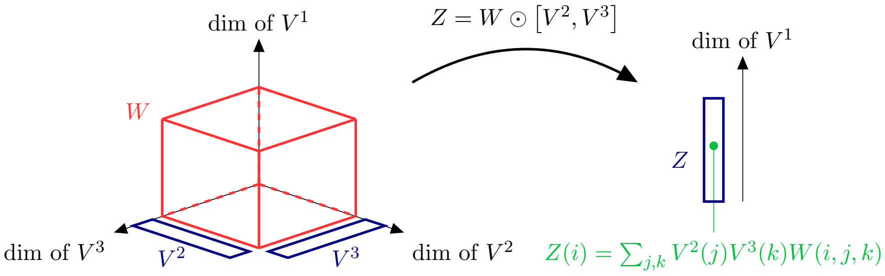

dim of V^1
$Z = W \odot \big[V^2, V^3\big]$
dim of V^1
W
Z
dim of V^3
V^2
V^3
dim of V^2
$Z(i) = \sum_{j,k} V^2(j)V^3(k)W(i, j, k)$

Figure 8: This figure illustrates the generalized inner product $Z = W \odot \big[V^2, V^3\big]$, where W is a cube of values illustrated in red with typical element $W(i, j, k)$. Also, the vectors Z and $V^i\ \forall i \in \{2, 3\}$ are drawn in blue along the dimension of the cube they correspond to.

Naming of the dimensions: Importantly, we should imagine that each side of W has a name, e.g., if W is a 3x2 matrix, then the i-th dimension of W could be named: "the dimension of V_i". This enables us to write: $Z^1 = W \odot V^1$ and $Z^2 = W \odot V^2$, where Z^1 is a 1x2 matrix (i.e., a vector with two elements) and Z^2 is a 3x1 matrix (i.e., a vector with three elements). The operator $\odot$ knows (thanks to the dimension name) that $W \odot V^1$ takes the weighted average w.r.t "the dimension of V_1", while $W \odot V^2$ must take the weighted average over "the dimension of V_2".

In the context of active inference, the matrix $\boldsymbol{A}$ has two dimensions that we could call "the observation dimension" (i.e., row-wise) and "the state dimension" (i.e., column-wise). Trivially, $\boldsymbol{A} \odot \boldsymbol{o}_\tau$ will then correspond to the average of $\boldsymbol{A}$ along the observation dimension and $\boldsymbol{A} \odot \hat{\boldsymbol{D}}_\tau$ will correspond to the average of $\boldsymbol{A}$ along the state dimension.

Appendix B: Generalized inner/outer products and other well-known products

In this section, we explore the relationship between our generalized inner and outer products—presented in appendix A—and other well known products in the literature.

Inner product of two vectors

The inner product of two vectors $\vec{a}$ and $\vec{b}$ of the same size is given by:

$$\vec{a} \cdot \vec{b} = \sum_{i=1}^{|\vec{a}|} a_i b_i, \tag{78}$$

where a_i and b_i are the elements of the vectors $\vec{a}$ and $\vec{b}$, respectively, and $|\vec{a}|$ is the number of elements in $\vec{a}$. This product is a special case of our generalised inner product, i.e.,

$$Z = W \odot V \Leftrightarrow Z = \sum_{i=1}^{|W|} W_i V_i, \tag{79}$$

where Z is a scalar (a 0-tensor), W and V are two vectors (two 1-tensors) of the same size, and $|W|$ is the number of elements in W.

Inner product of two matrices (Frobenius inner product)

The inner product of two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ of same sizes is given by:

$$\langle \boldsymbol{A}, \boldsymbol{B} \rangle = \sum_{i=1}^{|\boldsymbol{A}|_1} \sum_{j=1}^{|\boldsymbol{A}|_2} a_{ij} b_{ij}, \tag{80}$$

where $|\boldsymbol{A}|_1$ is the number of rows in $\boldsymbol{A}$, $|\boldsymbol{A}|_2$ is the number of columns in $\boldsymbol{A}$, and a_{ij} (b_{ij}) are the elements of the matrices $\boldsymbol{A}$ (respectively $\boldsymbol{B}$). This product is not a special case of our generalised inner product.

Inner product of two tensors (Frobenius inner product)

The inner product of two tensors $\boldsymbol{A}$ and $\boldsymbol{B}$ of same sizes is given by:

$$\langle \boldsymbol{A}, \boldsymbol{B} \rangle = \sum_{i_1=1}^{|\boldsymbol{A}|_1} \dots \sum_{i_n=1}^{|\boldsymbol{A}|_n} a(i_1, ..., i_n) b(i_1, ..., i_n), \tag{81}$$

where $a(i_1, ..., i_n)$ and $b(i_1, ..., i_n)$ are the elements of the tensors $\boldsymbol{A}$ and $\boldsymbol{B}$, respectively, and $|\boldsymbol{A}|_i$ is the number of elements in the i-th dimension of $\boldsymbol{A}$. This product is not a special case of our generalised inner product.

Standard matrix multiplication

Let $\boldsymbol{A}$ be an $n \times m$ matrix and $\vec{b}$ be a vector of size m. The standard matrix multiplication of $\boldsymbol{A}$ by $\vec{b}$ is given by:

$$\vec{c} = \boldsymbol{A}\vec{b} \Leftrightarrow \vec{c}_i = \sum_{j=1}^{m} \boldsymbol{A}_{ij} \vec{b}_j. \tag{82}$$

This is a special case of our generalised inner product, i.e.,

$$\vec{c} = \boldsymbol{A}\vec{b} = \boldsymbol{A} \odot \vec{b}. \tag{83}$$

Additionally, let $\boldsymbol{A}$ be an $n \times m$ matrix and $\vec{a}$ be a vector of size n, then:

$$\vec{c} = \boldsymbol{A}^T \vec{a} \Leftrightarrow \vec{c}_i = \sum_{j=1}^{n} \boldsymbol{A}_{ji} \vec{a}_j. \tag{84}$$

This is a special case of our generalised inner product, i.e.,

$$\vec{c} = \boldsymbol{A}^T \vec{a} = \boldsymbol{A} \odot \vec{a}. \tag{85}$$

Note that because the dimensions are "named" (c.f. Appendix A) the operator performs the transposition implicitly.

Outer product of two vectors

Given two vectors $\vec{a}$ and $\vec{b}$, there outer product—denoted $\vec{a} \otimes \vec{b}$—is a matrix defined as:

$$\vec{a} \otimes \vec{b} = \begin{bmatrix} a_1 b_1 & \dots & a_1 b_n \\ \vdots & \ddots & \vdots \\ a_m b_1 & \dots & a_m b_n \end{bmatrix}. \tag{86}$$

This outer product is a special case of our generalised outer product, where the operator is applied to only two vectors, i.e.

$$\vec{a} \otimes \vec{b} = \otimes \Big[\vec{a}, \vec{b} \Big]. \tag{87}$$

Outer product of two tensors

Given two tensors $\boldsymbol{U}$ and $\boldsymbol{V}$, the outer product of $\boldsymbol{U}$ and $\boldsymbol{V}$ is another tensor $\boldsymbol{W}$ such that:

$$\begin{aligned} \boldsymbol{W}(i_1, ..., i_n, j_1, ..., j_m) = \boldsymbol{V}(i_1, ..., i_n)\boldsymbol{U}(j_1, ..., j_m) \ \forall i_\alpha \in \{1, ..., |\boldsymbol{V}|_\alpha\} \forall \alpha \in \{1, ..., n\} \\ \forall j_\beta \in \{1, ..., |\boldsymbol{U}|_\beta\} \forall \beta \in \{1, ..., m\}. \end{aligned} \tag{88}$$

Given N vectors $V^i \ \forall i \in \{1, ..., N\}$, our outer product is a sequence of outer tensor products, i.e.,

$$\otimes \Big[V^1, ..., V^N \Big] = \Bigg[\Big[V^1 \otimes_{\text{tensor}} V^2 \Big] \otimes_{\text{tensor}} \cdots \Bigg] \otimes_{\text{tensor}} V^N, \tag{89}$$

where $\otimes_{\text{tensor}}$ and $\otimes$ are the tensor and generalised outer products, respectively.

Kronecker product

Given two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, the Kronecker product of $\boldsymbol{A}$ and $\boldsymbol{B}$—denoted $\otimes_K$—is a generalisation of the outer product from vectors to matrices defined as:

$$\boldsymbol{A} \otimes_K \boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B} & \dots & a_{1n}\boldsymbol{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\boldsymbol{B} & \dots & a_{mn}\boldsymbol{B} \end{bmatrix}, \tag{90}$$

where a_{ij} are the elements of $\boldsymbol{A}$. Note that even if the Kronecker product is a generalisation of the outer product, it is neither a special case nor a generalisation of our generalized outer product.

Hadamard product

Given two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ of the same size, the Hadamard product of $\boldsymbol{A}$ and $\boldsymbol{B}$ is an element-wise product defined by:

$$\boldsymbol{C} = \boldsymbol{A} \odot \boldsymbol{B} \Leftrightarrow c_{ij} = a_{ij} b_{ij} \ \forall i \in \{1, ..., |\boldsymbol{A}|_1\} \forall j \in \{1, ..., |\boldsymbol{A}|_2\}, \tag{91}$$

where a_{ij}, b_{ij} and c_{ij} are the elements in the i-th row and j-th column of $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$, respectively, $|\boldsymbol{A}|_1$ is the number of rows in $\boldsymbol{A}$, and $|\boldsymbol{A}|_2$ is the number of columns in $\boldsymbol{A}$. This product is unrelated to both our generalised inner and outer products.

Appendix C: Instance of variational message passing

This appendix provides a concrete instance of the method of Winn and Bishop discussed in Section 3. The generative model is as follows:

$$P(S, \boldsymbol{D}) = P(S|\boldsymbol{D}) P(\boldsymbol{D}) \tag{92}$$

where:

$$P(S|\boldsymbol{D}) = \text{Cat}(\boldsymbol{D}) \tag{93}$$

$$P(\boldsymbol{D}) = \text{Dir}(\boldsymbol{d}). \tag{94}$$

Additionally, the variational distribution is given by:

$$Q(\boldsymbol{D}) = \text{Dir}(\hat{\boldsymbol{d}}), \tag{95}$$

which means that we assume that S is an observed random variable. Let us start with the definition of the Dirichlet and categorical distributions written in the form of the exponential family:

$$\ln P(\boldsymbol{D}) = \underbrace{\begin{bmatrix} \boldsymbol{d}_1 - 1 \\ ... \\ \boldsymbol{d}_{|S|} - 1 \end{bmatrix}}_{\mu_D(\boldsymbol{d})} \cdot \underbrace{\begin{bmatrix} \ln \boldsymbol{D}_1 \\ ... \\ \ln \boldsymbol{D}_{|S|} \end{bmatrix}}_{u_D(\boldsymbol{D})} - \underbrace{\ln B(d)}_{z_D(\boldsymbol{d})} \tag{96}$$

$$\ln P(S|\boldsymbol{D}) = \underbrace{\begin{bmatrix} \ln \boldsymbol{D}_1 \\ ... \\ \ln \boldsymbol{D}_{|S|} \end{bmatrix}}_{\mu_S(\boldsymbol{D})} \cdot \underbrace{\begin{bmatrix} [S = 1] \\ ... \\ [S = |S|] \end{bmatrix}}_{u_S(S)} \tag{97}$$

where $\cdot$ performs an inner product of the two vectors it is applied to, $B(\boldsymbol{d})$ is the Beta function and $|S|$ is the number of values a state can take. The first step requires us to re-write Equation 97 as a function of $u_D(\boldsymbol{D})$, which is straightforward because $\mu_S(\boldsymbol{D})$ is just another name for $u_D(\boldsymbol{D})$. Using the fact that the inner product is commutative:

$$\ln P(S|\boldsymbol{D}) = \underbrace{\begin{bmatrix} [S = 1] \\ ... \\ [S = |S|] \end{bmatrix}}_{\mu_{S \to D}(S)} \cdot \underbrace{\begin{bmatrix} \ln \boldsymbol{D}_1 \\ ... \\ \ln \boldsymbol{D}_{|S|} \end{bmatrix}}_{u_D(\boldsymbol{D})} . \tag{98}$$

The second step aims to substitute (96) and (98) into the variational message passing equation (9), i.e.

$$\ln Q^*(\boldsymbol{D}) = \Big\langle \underbrace{\begin{bmatrix} \boldsymbol{d}_1 - 1 \\ ... \\ \boldsymbol{d}_{|S|} - 1 \end{bmatrix}}_{\mu_D(\boldsymbol{d})} \cdot \underbrace{\begin{bmatrix} \ln \boldsymbol{D}_1 \\ ... \\ \ln \boldsymbol{D}_{|S|} \end{bmatrix}}_{u_D(\boldsymbol{D})} - \underbrace{\ln B(\boldsymbol{d})}_{z_D(\boldsymbol{d})} \Big\rangle + \Big\langle \underbrace{\begin{bmatrix} [S = 1] \\ ... \\ [S = |S|] \end{bmatrix}}_{\mu_{S \to D}(S)} \cdot \underbrace{\begin{bmatrix} \ln \boldsymbol{D}_1 \\ ... \\ \ln \boldsymbol{D}_{|S|} \end{bmatrix}}_{u_D(\boldsymbol{D})} \Big\rangle + \text{Const}, \tag{99}$$

where $\langle \bullet \rangle$ refers to $\langle \bullet \rangle_{\sim Q_{\boldsymbol{D}}}$. Note that in the above equation, $\boldsymbol{d}_i$ are fixed parameters, therefore there is no posterior over d and the first expectation $\langle \cdot \rangle_{\sim Q_{\boldsymbol{D}}}$ can be removed. The third step rests on taking the exponential of both sides, using the linearity of expectation and factorising by $u_D(\boldsymbol{D})$ to obtain:

$$Q^*(\boldsymbol{D}) = \exp \left\{ \begin{bmatrix} \boldsymbol{d}_1 - 1 + \langle [S = 1] \rangle \\ ... \\ \boldsymbol{d}_{|S|} - 1 + \langle [S = |S|] \rangle \end{bmatrix} \cdot u_D(\boldsymbol{D}) + \text{Const} \right\}, \tag{100}$$

where $z_D(\boldsymbol{d})$ have been absorbed into the constant term because it does not depend on $\boldsymbol{D}$. The fourth step is a re-parameterisation done by observing that $\langle [S = i] \rangle$ is the i-th element of the expectation of the vector $u_S(S)$, i.e. $\langle u_S(S) \rangle_i = \langle [S = i] \rangle$:

$$Q^*(\boldsymbol{D}) = \exp\left\{\underbrace{\begin{bmatrix} \boldsymbol{d}_1 - 1 + \langle u_S(S)\rangle_1 \\ \dots \\ \boldsymbol{d}_{|S|} - 1 + \langle u_S(S)\rangle_{|S|} \end{bmatrix}}_{\tilde{\mu}_D(\dots)+\tilde{\mu}_{S\to D}(\dots)} \cdot u_D(\boldsymbol{D}) + \text{Const}\right\}. \tag{101}$$

The last step consists of computing the expectation of $\langle u_S(S)\rangle_i$ for all i. This can be achieved by realising that the probability of an indicator function for an event is the probability of this event, i.e $\langle u_S(S)\rangle_i = \langle [S = i]\rangle = Q(S = i) = \hat{\boldsymbol{D}}_i$ where $\hat{\boldsymbol{D}}$ is a one hot representation of the observed value for S. Substituting this result in Equation 101, leads to the final result:

$$Q^*(\boldsymbol{D}) = \exp\left\{\begin{bmatrix} \boldsymbol{d}_1 - 1 + \hat{\boldsymbol{D}}_1 \\ \dots \\ \boldsymbol{d}_{|S|} - 1 + \hat{\boldsymbol{D}}_{|S|} \end{bmatrix} \cdot u_D(\boldsymbol{D}) + \text{Const}\right\}. \tag{102}$$

Indeed, the above equation is in fact a Dirichlet distribution in exponential family form, and can be re-written into its usual form to obtain the final update equation:

$$Q^*(\boldsymbol{D}) = \text{Dir}(\boldsymbol{d} + \hat{\boldsymbol{D}}). \tag{103}$$

Appendix D: Expected log of Dirichlet distribution

Definition 7 *A probability distribution over $\boldsymbol{x}$ parameterized by $\boldsymbol{\mu}$ is said to belong to the exponential family if its probability mass function $P(\boldsymbol{x}|\boldsymbol{\mu})$ can be written as:*

$$P(\boldsymbol{x}|\boldsymbol{\mu}) = h(\boldsymbol{x}) \exp\Big[\boldsymbol{\mu} \cdot T(\boldsymbol{x}) - A(\boldsymbol{\mu})\Big], \tag{104}$$

where $h(\boldsymbol{x})$ is the base measure, $\boldsymbol{\mu}$ is the vector of natural parameters, $T(\boldsymbol{x})$ is the vector of sufficient statistics, and $A(\boldsymbol{\mu})$ is the log partition.

Lemma 8 *The log partition is given by:*

$$A(\boldsymbol{\mu}) = \ln \int h(\boldsymbol{x}) \exp\Big[\boldsymbol{\mu} \cdot T(\boldsymbol{x})\Big] d\boldsymbol{x}. \tag{105}$$

Proof Starting with the fact that $P(\boldsymbol{x}|\boldsymbol{\mu})$ integrate to one:

$$\int P(\boldsymbol{x}|\boldsymbol{\mu})d\boldsymbol{x} = \int h(\boldsymbol{x}) \exp\Big[\boldsymbol{\mu} \cdot T(\boldsymbol{x}) - A(\boldsymbol{\mu})\Big] d\boldsymbol{x} = 1 \tag{106}$$

$$\Leftrightarrow \quad \frac{1}{\exp A(\boldsymbol{\mu})} \int h(\boldsymbol{x}) \exp\Big[\boldsymbol{\mu} \cdot T(\boldsymbol{x})\Big] d\boldsymbol{x} = 1 \tag{107}$$

$$\Leftrightarrow \quad A(\boldsymbol{\mu}) = \ln \int h(\boldsymbol{x}) \exp\Big[\boldsymbol{\mu} \cdot T(\boldsymbol{x})\Big] d\boldsymbol{x} \tag{108}$$

■

Lemma 9 *The gradient of the log partition function is the expectation of the sufficient statistics, i.e.,*

$$\frac{\partial A(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}} = \mathbb{E}_{P(\boldsymbol{x}|\boldsymbol{\mu})}\Big[T(\boldsymbol{x})\Big]. \tag{109}$$

Proof Restarting with the derivative of the result of Lemma 8:

$$\frac{\partial A(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}} = \frac{\partial}{\partial \boldsymbol{\mu}}\left[\ln \int h(\boldsymbol{x}) \exp\Big[\boldsymbol{\mu} \cdot T(\boldsymbol{x})\Big] d\boldsymbol{x}\right], \tag{110}$$

and using the chain rule:

$$\frac{\partial A(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}} = \frac{1}{\int h(\boldsymbol{x}) \exp\big[\boldsymbol{\mu} \cdot T(\boldsymbol{x})\big] d\boldsymbol{x}} \frac{\partial}{\partial \boldsymbol{\mu}}\left[\int h(\boldsymbol{x}) \exp\Big[\boldsymbol{\mu} \cdot T(\boldsymbol{x})\Big] d\boldsymbol{x}\right]. \tag{111}$$

Note that the denominator of the first term is equal to the exponential of $A(\boldsymbol{\mu})$, and we can swap the derivative and the integral because the limit of integration does not depend on the parameters $\boldsymbol{\mu}$:

$$\frac{\partial A(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}} = \frac{1}{\exp A(\boldsymbol{\mu})} \int \frac{\partial}{\partial \boldsymbol{\mu}}\Big[h(\boldsymbol{x}) \exp\Big[\boldsymbol{\mu} \cdot T(\boldsymbol{x})\Big]\Big] d\boldsymbol{x} \tag{112}$$

$$= \frac{1}{\exp A(\boldsymbol{\mu})} \int h(\boldsymbol{x}) \frac{\partial}{\partial \boldsymbol{\mu}}\Big[\exp\Big[\boldsymbol{\mu} \cdot T(\boldsymbol{x})\Big]\Big] d\boldsymbol{x}. \tag{113}$$

Using the chain rule again:

$$\frac{\partial A(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}} = \frac{1}{\exp A(\boldsymbol{\mu})} \int h(\boldsymbol{x}) \exp\Big[\boldsymbol{\mu} \cdot T(\boldsymbol{x})\Big] T(\boldsymbol{x}) d\boldsymbol{x} \tag{114}$$

$$= \int h(\boldsymbol{x}) \exp\Big[\boldsymbol{\mu} \cdot T(\boldsymbol{x}) - A(\boldsymbol{\mu})\Big] T(\boldsymbol{x}) d\boldsymbol{x} \tag{115}$$

$$= \int P(\boldsymbol{x}|\boldsymbol{\mu}) T(\boldsymbol{x}) d\boldsymbol{x} \tag{116}$$

$$= \mathbb{E}_{P(\boldsymbol{x}|\boldsymbol{\mu})}\Big[T(\boldsymbol{x})\Big]. \tag{117}$$

$\blacksquare$

Theorem 10 *If $\boldsymbol{D}$ is distributed according to a Dirichlet distribution $Q(\boldsymbol{D}) = Dir(\boldsymbol{D}; \hat{\boldsymbol{d}})$, then:* $\mathring{\boldsymbol{D}} = \mathbb{E}_{Q(\boldsymbol{D})}\big[\ln \boldsymbol{D}\big] \Leftrightarrow \mathring{\boldsymbol{D}}_i = \psi(d_i) - \psi(\sum_j d_j)$.

Proof Let $\boldsymbol{\mu}$ be equal to $\hat{\boldsymbol{d}} - \vec{1}$. Taking the exponential of both sides in Equation 96 and using that $\hat{\boldsymbol{d}} = \boldsymbol{\mu} + \vec{1}$, we obtain:

$$Q(\boldsymbol{D}) = \exp\left\{ \underbrace{\begin{bmatrix} \hat{\boldsymbol{d}}_1 - 1 \\ ... \\ \hat{\boldsymbol{d}}_{|S|} - 1 \end{bmatrix}}_{\boldsymbol{\mu}} \cdot \underbrace{\begin{bmatrix} \ln \boldsymbol{D}_1 \\ ... \\ \ln \boldsymbol{D}_{|S|} \end{bmatrix}}_{T(\boldsymbol{D})} - \underbrace{\ln B(\boldsymbol{\mu} + \vec{1})}_{A(\boldsymbol{\mu})} \right\}, \tag{118}$$

where $\boldsymbol{\mu}$ is the vector of natural parameters, $T(\boldsymbol{D})$ is the vector of sufficient statistics, $A(\boldsymbol{\mu})$ is the log partition, and $B(\bullet)$ is the beta function. Using the result of Lemma 9:

$$\mathring{\boldsymbol{D}} \triangleq \mathbb{E}_{Q(\boldsymbol{D})}\Big[T(\boldsymbol{D})\Big] = \mathbb{E}_{Q(\boldsymbol{D})}\Big[\ln \boldsymbol{D}\Big] = \frac{\partial A(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}} = \frac{\partial}{\partial \boldsymbol{\mu}}\Big[\ln B(\boldsymbol{\mu} + \vec{1})\Big]. \tag{119}$$

We now focus on a typical element of $\mathring{\boldsymbol{D}}$:

$$\mathring{\boldsymbol{D}}_i = \frac{\partial}{\partial \mu_i}\Big[\ln B(\boldsymbol{\mu} + \vec{1})\Big], \tag{120}$$

and use the definition of the beta function:

$$\begin{aligned} \mathring{\boldsymbol{D}}_i &= \frac{\partial}{\partial \boldsymbol{\mu}_i}\left[\ln \frac{\prod_k \Gamma(\boldsymbol{\mu}_k + 1)}{\Gamma(\sum_k \boldsymbol{\mu}_k + 1)}\right] && (121) \\ &= \frac{\partial}{\partial \boldsymbol{\mu}_i}\left[\sum_k \ln \Gamma(\boldsymbol{\mu}_k + 1) - \ln \Gamma(\textstyle\sum_k \boldsymbol{\mu}_k + 1)\right] && (122) \\ &= \frac{\partial}{\partial \boldsymbol{\mu}_i}\Big[\ln \Gamma(\boldsymbol{\mu}_i + 1)\Big] - \frac{\partial}{\partial \boldsymbol{\mu}_i}\Big[\ln \Gamma(\textstyle\sum_k \boldsymbol{\mu}_k + 1)\Big], && (123) \end{aligned}$$

where $\Gamma(\bullet)$ is the gamma function. The last step relies on the chain rule:

$$\begin{aligned}
\mathring{\boldsymbol{D}}_i &= \frac{\partial}{\partial(\boldsymbol{\mu}_i + 1)}\Big[\ln\Gamma(\boldsymbol{\mu}_i + 1)\Big]\underbrace{\frac{\partial\boldsymbol{\mu}_i + 1}{\partial\boldsymbol{\mu}_i}}_{=1} - \frac{\partial}{\partial(\sum_k \boldsymbol{\mu}_k + 1)}\Big[\ln\Gamma(\textstyle\sum_k \boldsymbol{\mu}_k + 1)\Big]\underbrace{\frac{\partial\sum_k \boldsymbol{\mu}_k + 1}{\partial\boldsymbol{\mu}_i}}_{=1} && (124)\\
&= \frac{\partial}{\partial(\boldsymbol{\mu}_i + 1)}\Big[\ln\Gamma(\boldsymbol{\mu}_i + 1)\Big] - \frac{\partial}{\partial(\sum_k \boldsymbol{\mu}_k + 1)}\Big[\ln\Gamma(\textstyle\sum_k \boldsymbol{\mu}_k + 1)\Big] && (125)\\
&= \psi(\boldsymbol{\mu}_i + 1) - \psi(\textstyle\sum_k \boldsymbol{\mu}_k + 1) && (126)\\
&= \psi(\hat{\boldsymbol{d}}_i) - \psi(\textstyle\sum_k \hat{\boldsymbol{d}}_k), && (127)
\end{aligned}$$

where we used that $\hat{\boldsymbol{d}} = \boldsymbol{\mu} + \vec{1}$ and the definition of the digamma function, i.e., $\psi(x) = \frac{\partial \ln \Gamma(x)}{\partial x}$. ∎

Appendix E: Relationship between BTAI and active inference (Lemmas)

Lemma 11 *Under the assumption that the probability of observations and states are independent of future actions, i.e.,*

$$\forall j \in \mathbb{N}_{>0},\ Q(O_{t+i}|\pi_i) \approx Q(O_{t+i}|\pi_{i+j}) \text{ and } Q(S_{t+i}|\pi_i) \approx Q(S_{t+i}|\pi_{i+j}), \tag{128}$$

then:

$$\forall j \in \mathbb{N}_{>0},\ \boldsymbol{G}(\pi_{i+j}, t+i) \approx \boldsymbol{G}(\pi_i, t+i). \tag{129}$$

Proof The proof is straightforward, we start with the following definition:

$$\boldsymbol{G}(\pi_{i+j}, t+i) = D_{\mathrm{KL}}[Q(O_{t+i}|\pi_{i+j})||P(O_{t+i})] \ + \ \mathbb{E}_{Q(S_{t+i}|\pi_{i+j})}[\mathrm{H}[P(O_{t+i}|S_{t+i})]]. \tag{130}$$

Then, using the assumption that the probability of observations and states are independent of future actions, i.e., $\forall j \in \mathbb{N}_{>0},\ Q(O_{t+i}|\pi_{i+j}) \approx Q(O_{t+i}|\pi_i)$ and $\forall j \in \mathbb{N}_{>0},\ Q(S_{t+i}|\pi_{i+j}) \approx Q(S_{t+i}|\pi_i)$, we get:

$$\boldsymbol{G}(\pi_{i+j}, t+i) \approx D_{\mathrm{KL}}[Q(O_{t+i}|\pi_i)||P(O_{t+i})] \ + \ \mathbb{E}_{Q(S_{t+i}|\pi_i)}[\mathrm{H}[P(O_{t+i}|S_{t+i})]] \triangleq \boldsymbol{G}(\pi_i, t+i). \tag{131}$$

∎

Lemma 12 *The aggregated cost for an arbitrary N is given by:*

$$\boldsymbol{G}^{aggre}_{\pi_N} = \sum_{i=1}^{N} \boldsymbol{G}(\pi_i, t+i), \tag{132}$$

Proof The proof is done by induction. The initialisation holds for $N = 1$, indeed, $\pi_1 = \{U_t\}$ and by definition:

$$\underbrace{\boldsymbol{G}(\pi_1)}_{\boldsymbol{G}^{aggre}_{\pi_1}} = \sum_{\tau=t+1}^{t+1} \boldsymbol{G}(\pi_1, \tau) = \underbrace{D_{\mathrm{KL}}[Q(O_{t+1}|\pi_1)||P(O_{t+1})] \; + \; \mathbb{E}_{Q(S_{t+1}|\pi_1)}[\mathrm{H}[P(O_{t+1}|S_{t+1})]]}_{\boldsymbol{G}^{local}_{\pi_1}}, \tag{133}$$

$$\Leftrightarrow \boldsymbol{G}^{aggre}_{\pi_1} = \boldsymbol{G}^{aggre}_{\pi_0} + \boldsymbol{G}^{local}_{\pi_1}, \tag{134}$$

because by definition $\boldsymbol{G}^{aggre}_{\pi_0} = 0$. Then, assuming that $\boldsymbol{G}^{aggre}_{\pi_N} = \sum_{i=1}^{N} \boldsymbol{G}(\pi_i, t+i)$ holds for some N, we show that its hold for $N+1$ as well. By definition:

$$\boldsymbol{G}^{aggre}_{\pi_{N+1}} = \boldsymbol{G}^{aggre}_{\pi_N} + \boldsymbol{G}^{local}_{\pi_{N+1}}, \tag{135}$$

and:

$$\boldsymbol{G}^{local}_{\pi_{N+1}} = \boldsymbol{G}(\pi_{N+1}, t+N+1). \tag{136}$$

Using the inductive hypothesis and the above two equations:

$$\begin{aligned} \boldsymbol{G}^{aggre}_{\pi_{N+1}} &= \sum_{i=1}^{N} \boldsymbol{G}(\pi_i, t+i) + \boldsymbol{G}(\pi_{N+1}, t+N+1) \qquad (137) \\ &= \sum_{i=1}^{N+1} \boldsymbol{G}(\pi_i, t+i). \qquad (138) \end{aligned}$$

■

Appendix F: Notation

In this appendix, we introduce the notation used throughout this paper. The following sub-sections describe the notation related to sets of numbers, tensors, probability distributions, global variables, multi-indices and random variables, respectively.

Sets of numbers

Definition 13 *Let $\mathbb{N}_{>0}$ be the set of all strictly positive integers defined as:*

$$\mathbb{N}_{>0} = \{x \in \mathbb{N} \mid x > 0\}, \tag{139}$$

where $\mathbb{N}$ is the set of all natural numbers.

Definition 14 *Let $\mathbb{R}_{>0}$ be the set of all strictly positive real numbers defined as:*

$$\mathbb{R}_{>0} = \{x \in \mathbb{R} \mid x > 0\}, \tag{140}$$

where $\mathbb{R}$ is the set of all real numbers.

Tensors

Definition 15 *An **n-tensor** is an n-dimensional array of values. Each element of an n-tensor is indexed by an n-tuple of non-negative integers, i.e., $(x_1, ..., x_n)$ where $x_i \in \mathbb{N}_{>0}\ \forall i \in \{1, 2, ..., n\}$.*

Definition 16 *Let T be an n-tensor. The **element** of T indexed by the n-tuple $(x_1, ..., x_n)$ is a real number denoted by $T(x_1, ..., x_n)$.*

Example 1 *Let T be a 2-tensor defined as:*

$$T = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}. \tag{141}$$

Then $T(1, 1) = 1$, $T(1, 2) = 2$, $T(2, 1) = 3$ and $T(2, 2) = 4$.

Remark 17 *A 0-tensor is a scalar, a 1-tensor is a vector, and a 2-tensor is a matrix.*

Definition 18 *Let T be an n-tensor. The **size** of T is a vector of size n denoted $|T|$ whose i-th element corresponds to the size of the i-th dimension of T.*

Example 2 *Let T be a 2-tensor defined as:*

$$T = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}. \tag{142}$$

Then $|T|_1 = 2$ and $|T|_2 = 3$.

Definition 19 *Let T be an n-tensor. A **1-sub-tensor** of T is a 1-tensor obtained by selecting a 1-dimensional slice of T, i.e.,*

$$T(x_1, ..., x_{i-1}, \bullet, x_{i+1}, , ..., x_n), \tag{143}$$

where $\bullet$ represents the selection of a 1-dimensional slice of T, and the values of all $x_{j \neq i}$ must be set to specific values in $\{1, ..., |T|_j\}$. Figure 9 (left) illustrates the notion of a 1-dimensional slice.

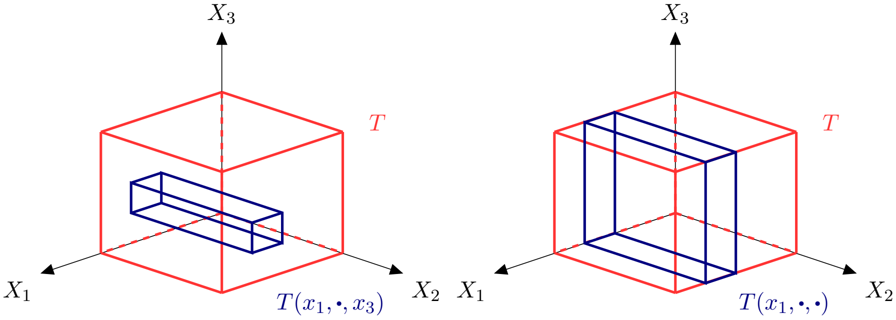


Figure 9: This figure illustrates the notion of a 1-dimensional slice (on the left) and of a 2-dimensional slice (on the right).

Definition 20 *Let T be an n-tensor and $m < n$. An **m-sub-tensor** of T (denoted W) is an m-tensor obtained by selecting an m-dimensional slice of T, i.e.,*

$$W = T(x_1, ..., x_{i_1-1}, \bullet, x_{i_1+1}, ..., x_{i_m-1}, \bullet, x_{i_m+1}, ..., x_n), \tag{144}$$

where $i_k \in \{1, ..., n\}$ $\forall k \in \{1, ..., m\}$ are indices representing the dimension being selected. Naturally, the k-th dimension of W corresponds to the i_k-th dimension of T for $k \in \{1, ..., m\}$. Importantly, the sextuple of dots in the middle of the expression represents that there will be m symbols "$\bullet$", i.e., one for each dimension selected. Figure 9 (right) illustrates the special case of a 2-dimensional slice, i.e., $m = 2$.

Example 3 *Let T be a 3-tensor such that:*

- $|T|_1 = 2$

- $T(1, x_2, x_3) = 1, \ \forall (x_2, x_3) \in \{1, ..., |T|_2\} \times \{1, ..., |T|_3\}$

- $T(2, x_2, x_3) = 2, \ \forall (x_2, x_3) \in \{1, ..., |T|_2\} \times \{1, ..., |T|_3\}$.

Then $T(1, \bullet, \bullet)$ is a 2-sub-tensor of T full of ones, and $T(2, \bullet, \bullet)$ is a 2-sub-tensor of T full of twos.

Probability distributions

Definition 21 *A* ***random n-tensor*** *is an n-tensor over which we have an n-dimensional probability distribution.*

Remark 22 *A random variable is a random 0-tensor, a random vector is a random 1-tensor, and a random matrix is a random 2-tensor.*

Definition 23 *An n-tensor T is said to* ***represent a joint distribution*** *over a set of n random variables $\{X_1, ..., X_n\}$ if:*

$$P(X_1 = x_1, ..., X_n = x_n) = T(x_1, ..., x_n). \tag{145}$$

For conciseness, if T represents $P(X_1, ..., X_n)$ we let:

$$P(X_1, ..., X_n) = Cat(T). \tag{146}$$

Remark 24 *If T represents $P(X_1, ..., X_n)$, then the sum of its elements must equal one.*

Remark 25 *In contrast, if T is a* ***random*** *n-tensor, then:*

$$P(X_1, ..., X_n|T) = Cat(T), \tag{147}$$

which means that the joint probability over $X_1, ..., X_n$ is represented by T, and because T is a ***random*** *tensor (taking values in the set of valid n-tensors $\mathbb{T}_n$, i.e., the set of all n-tensors whose elements sum up to one), we must specify which instance of $T \in \mathbb{T}_n$ should be used to define the joint distribution over $X_1, ..., X_n$.*

Definition 26 *An m-tensor R is said to* ***represent a conditional distribution*** *over a set of m random variables $\{X_1, ..., X_m\}$ if:*

$$P(X_1 = x_1, ..., X_n = x_n | X_{n+1} = x_{n+1}, ..., X_m = x_m) = R(x_1, ..., x_m). \tag{148}$$

For conciseness, if R represents $P(X_1, ..., X_n|X_{n+1}, ..., X_m)$ we let:

$$P(X_1, ..., X_n|X_{n+1}, ..., X_m) = Cat(R). \tag{149}$$

Remark 27 *If R represents $P(X_1, ..., X_n|X_{n+1}, ..., X_m)$, then the elements of the n-sub-tensor $R(\bullet, ..., \bullet, x_{n+1}, ..., x_m)$ must sum to one $\forall x_i \in \{1, ..., |R|_i\}\ \forall i \in \{n+1, ..., m\}$.*

Remark 28 *If R is a random m-tensor, then:*

$$P(X_1, ..., X_n | X_{n+1}, ..., X_m, R) = Cat(R). \tag{150}$$

Remark 29 *Importantly, definition 26 uses the symbol T to represent $P(X_1, ..., X_n)$ and definition 23 uses the symbol R to represent $P(X_1, ..., X_n | X_{n+1}, ..., X_m)$. Throughout this document, different symbols will be used for representing joint and conditional distributions.*

Definition 30 *Let R be a random m-tensor representing $P(X_1, ..., X_n | X_{n+1}, ..., X_m, R)$ and $k = m - n$ be the number of variables upon which the variables $X_1, ..., X_n$ are conditioned. Having a **Dirichlet prior** over R means that:*

$$P(R) = \prod_{i_1=1}^{|R|_{n+1}} ... \prod_{i_k=1}^{|R|_m} Dir\Big(\boldsymbol{r}(i_1, ..., i_k, \bullet)\Big), \tag{151}$$

where $\boldsymbol{r}$ is an $(m+1)$-tensor such that the 1-sub-tensor $\boldsymbol{r}(i_1, ..., i_k, \bullet)$ contains the parameters of the Dirichlet prior over $P(X_1, ..., X_n | X_{n+1} = i_1, ..., X_m = i_k)$. For conciseness, we denote the prior over R as:

$$P(R) = Dir(\boldsymbol{r}). \tag{152}$$

Remark 31 *Definition 30 implicitly means that if V is a 1-tensor then $Dir(V)$ represents a Dirichlet distribution. However, if V is an m-tensor (with $m \neq 1$) then $Dir(V)$ represents a product of Dirichlet distributions.*

Remark 32 *If R is a random m-tensor representing $P(X_1, ..., X_n | X_{n+1}, ..., X_m, R)$, then its prior will be a product of $|X_{n+1}| \times ... \times |X_m|$ Dirichlet distributions, where $|X_i|$ is the number of values that X_i can take. Additionally, each Dirichlet distribution will have $|X_1| \times ... \times |X_n|$ parameters stored in the last dimension of $\boldsymbol{r}$, where $\boldsymbol{r}$ is the tensor storing the parameters of the prior over R, i.e. $P(R) = Dir(\boldsymbol{r})$.*

Example 4 *Let $\boldsymbol{A}$ be a random 2-tensor representing $P(O|S, \boldsymbol{A})$, then the Dirichlet prior over $\boldsymbol{A}$ is given by:*

$$P(\boldsymbol{A}) = Dir(\boldsymbol{a}) \triangleq \prod_{i=1}^{|\boldsymbol{A}|_2} Dir\Big(\boldsymbol{a}(i, \bullet)\Big), \tag{153}$$

where $|\boldsymbol{A}|_2$ is the number of values that S can take (i.e., the number of hidden states).

Example 5 *Let $\boldsymbol{B}$ be a random 3-tensor representing $P(S_{\tau+1}|S_\tau, U_\tau, \boldsymbol{B})$, then the Dirichlet prior over $\boldsymbol{B}$ is given by:*

$$P(\boldsymbol{B}) = Dir(\boldsymbol{b}) \triangleq \prod_{i_1=1}^{|\boldsymbol{B}|_2} \prod_{i_2=1}^{|\boldsymbol{B}|_3} Dir\Big(\boldsymbol{b}(i_1, i_2, \bullet)\Big), \tag{154}$$

where $|\boldsymbol{B}|_2$ is the number of values that S_τ can take (i.e., the number of hidden states) and $|\boldsymbol{B}|_3$ is the number of values that U_τ can take (i.e., the number of actions).

Global labels

Definition 33 *The **number of actions** available to the agent is denoted $|U|$.*

Definition 34 *The **number of states** in the environment is denoted $|S|$.*

Definition 35 *The **number of observations** that the agent can make is denoted $|O|$.*

Definition 36 *The **number of policies** that the agent can pick from is denoted $|\pi|$.*

Definition 37 *The time point representing the **present** is a natural number denoted t.*

Definition 38 *The **time-horizon** (i.e., the time point after which the agent stops modelling the sequence of hidden states) is a natural number denoted T.*

Multi-indices

Definition 39 *A **multi-index** is a sequence of indices denoted by:*

$$I = (i_1, ..., i_n), \tag{155}$$

where $i_j \in \mathcal{D}\ \forall j \in \{1, ..., n\}$, and in this paper $\mathcal{D} = \{1, ..., |U|\}$.

Remark 40 *Multi-indices are used to index random variables such that S_I is the hidden state obtained after taking the sequence of actions described by I, and O_I is the random variable representing the observation generated by S_I.*

Definition 41 *The **last index** of a multi-index is denoted I_{last}, i.e., I_{last} is the last element of the sequence I.*

Definition 42 *The **one-hot representation** of the action corresponding to I_{last} is denoted $\vec{I}_{last}$.*

Definition 43 *Given a multi-index I, $I \setminus last$ corresponds to the sequence of actions described by **I without the last element**.*

Remark 44 *In Section 6, when a hidden state (i.e., S_I) is indexed by I, then $S_{I \backslash last}$ will be the parent of S_I.*

Definition 45 *Given an expandable generative model, $\mathbb{I}_t$ is the set of* ***all multi-indices already expanded*** *from the current state S_t.*

Remark 46 *In Section 6 each time a hidden state (i.e., S_I) is added to the generative model, I is added to the set of all multi-indices already expanded $\mathbb{I}_t$.*

Random variables and parameters of their distributions

Remark 47 *Parameters of the posterior distributions are recognizable by the hat notation, e.g., $\hat{\boldsymbol{a}}$, $\hat{\boldsymbol{b}}$ and $\hat{\boldsymbol{d}}$ will be posterior parameters, while $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{d}$ will be prior parameters.*

Remark 48 *The expected logarithm of an arbitrary tensor $\boldsymbol{X}$ representing a conditional or a joint distribution is denoted $\bar{\boldsymbol{X}}$, e.g. $\bar{\boldsymbol{A}} = \mathbb{E}_{Q(\boldsymbol{A})}[\ln \boldsymbol{A}]$ and $\bar{\boldsymbol{B}} = \mathbb{E}_{Q(\boldsymbol{B})}[\ln \boldsymbol{B}]$.*

Definition 49 *Let U_τ be a random variable taking values in $\{1, ..., |U|\}$ indexing all possible actions. The prior distribution over U_τ is a categorical distribution represented by $\boldsymbol{\Theta}_\tau$. The posterior distribution over U_τ is a categorical distribution represented by $\hat{\boldsymbol{\Theta}}_\tau$.*

Definition 50 *Let S_τ be a random variable taking values in $\{1, ..., |S|\}$ indexing all possible states. The prior and posterior distributions over S_τ are categorical distributions represented by different tensors depending on the generative model being considered. Therefore, those distributions are defined in Sections 4 and 6.*

Definition 51 *Let O_τ be an observed random variable taking values in $\{1, ..., |O|\}$ indexing all possible observations. The prior distribution over O_τ is a categorical distribution conditioned on S_τ and represented by the random matrix $\boldsymbol{A}$. There is no posterior distribution over O_τ because O_τ is observed, i.e., realised.*

Definition 52 *Let O_I be a random variable taking values in $\{1, ..., |O|\}$ indexing all possible observations. The prior distribution over O_I is a categorical distribution conditioned on S_I and represented by the random matrix $\bar{\boldsymbol{A}}$. Note that O_I refers to an observation in the future and is therefore a hidden variable. The posterior distribution over O_I is a categorical distribution represented by $\hat{\boldsymbol{E}}_I$.*

Definition 53 *Let S_I be an random variable taking values in $\{1, ..., |S|\}$ indexing all possible states. The prior distribution over S_I is a categorical distribution conditioned on $S_{I \backslash last}$ and represented by the matrix $\bar{\boldsymbol{B}}_I \triangleq \bar{\boldsymbol{B}}(\bullet, \bullet, I_{last})$. The posterior distribution over S_I is a categorical distribution represented by $\hat{\boldsymbol{D}}_I$.*

Definition 54 *Let $\boldsymbol{A}$ be a $|O| \times |S|$ random matrix defining the probability of an observation O_τ given the hidden state S_τ. The prior distribution over $\boldsymbol{A}$ is a product of Dirichlet distributions whose parameters are stored in a*

$|S| \times |O|$ matrix $\boldsymbol{a}$. The posterior distributions over $\boldsymbol{A}$ is also a product of Dirichlet distribution, but the parameters are stored in a $|S| \times |O|$ matrix $\hat{\boldsymbol{a}}$.

Definition 55 *Let $\boldsymbol{B}$ be a $|S| \times |S| \times |U|$ random 3-tensor defining the probability of transiting from S_τ to $S_{\tau+1}$ when taking action U_τ. The prior distribution over $\boldsymbol{B}$ is a product of Dirichlet distributions whose parameters are stored in a $|S| \times |U| \times |S|$ 3-tensor $\boldsymbol{b}$. The posterior distribution over $\boldsymbol{B}$ is also a product of Dirichlet distributions, but the parameters are stored in a $|S| \times |U| \times |S|$ 3-tensor $\hat{\boldsymbol{b}}$.*

Definition 56 *Let $\boldsymbol{D}$ be a random vector of size $|S|$ defining the probability of the initial state S_0. The prior distribution over $\boldsymbol{D}$ is a Dirichlet distribution whose parameters are stored in a vector $\boldsymbol{d}$ of size $|S|$. The posterior distribution over $\boldsymbol{D}$ is also a Dirichlet distribution, but the parameters are stored in a vector $\hat{\boldsymbol{d}}$ of size $|S|$.*

Definition 57 *Let $\boldsymbol{\Theta}_\tau$ $\forall \tau \in \{0, ..., t-1\}$ be a random vector of size $|U|$ defining the probability of the action U_τ. The prior distribution over $\boldsymbol{\Theta}_\tau$ is a Dirichlet distribution whose parameters are stored in a vector $\boldsymbol{\theta}_\tau$ of size $|U|$. The posterior distribution over $\boldsymbol{\Theta}_\tau$ is also a Dirichlet distribution, but the parameters are stored in a vector $\hat{\boldsymbol{\theta}}_\tau$ of size $|U|$.*

Definition 58 *Let γ be a random variable taking values in $\mathbb{R}_{>0}$. The prior distribution over γ is a gamma distribution with shape parameter $\boldsymbol{\alpha} = 1$ and rate parameter $\boldsymbol{\beta} \in \mathbb{R}_{>0}$. The posterior distribution over γ is a gamma distribution with shape parameter $\hat{\boldsymbol{\alpha}} = 1$ and rate parameter $\hat{\boldsymbol{\beta}} \in \mathbb{R}_{>0}$.*

Definition 59 *Let π be a random variable taking values in $\{1, ..., |\pi|\}$ indexing all possible policies. The prior distribution over π is a softmax function of the vector $\boldsymbol{G}$ multiplied by minus the precision γ. Note that $\boldsymbol{G}$ is a vector of size $|\pi|$ whose i-th element is the expected free energy of the i-th policy. The posterior distribution over π is a categorical distribution whose parameters are stored in a vector $\hat{\boldsymbol{\pi}}$ of size $|\pi|$.*

Appendix G: Multi-armed bandit problem

In the multi-armed bandit problem, the agent is prompted with K actions (one for each bandit's arm). Pulling the i-th arm returns a reward sampled from the reward distribution $P_i(X)$ associated to this arm. Let μ_i be the mean of the i-th reward distribution and $T_i(n)$ be the number of times the i-th bandit has been selected after n plays. To solve the bandit problem, one needs to come up with an *allocation strategy* that selects the action that minimises the agent's regret defined as:

$$R_n = \mu^* n - \sum_{i=1}^{K} \mu_i \mathbb{E}[T_i(n)], \tag{156}$$

where μ^* is the average reward of the best action. Note that an upper bound of $\mathbb{E}[T_i(n)]$ is derived by first upper bounding $T_i(n)$, and then using: the Chernoff-Hoeffing bound, the Bernstein inequality and some properties of p-series, c.f., proof of Theorem 1 in Auer et al (2002) for details. So, if we first assume that,

$$UCB1_i = \underbrace{\quad \bar{X}_i \quad}_{\text{exploitation}} + \underbrace{\sqrt{\frac{2\ln n}{n_i}}}_{\text{exploration}}, \tag{157}$$

where n_i is the number of times the i-th action has been selected, and $\bar{X}_i$ is the average reward received after taking the i-th action. Then, the main result of Auer et al (2002) was to show that if an allocation strategy was using the UCB1 criterion to select the next action, the expected regret of this allocation strategy will grow at most logarithmically in the number of plays n, i.e., $\mathcal{O}(\ln n)$. In addition, since it is known that the expected regret of the (best) allocation strategy grows at least logarithmically in n (Lai and Robbins, 1985), we say that the UCB1 criterion resolves the exploration / exploitation trade-off, i.e., the UCB1 criterion ensures that the expected regret grows as slowly as possible.

Appendix H: The exponential complexity class

In this appendix, we precisely pinpoint the exponential complexity class that is addressed in this paper, but first, we introduce a multi-index notation. Multi-indices will help us to refer to hidden states in the future. Naturally enough the indexes inside the multi-indices will correspond to the actions the agent will have to perform to reach the hidden state, e.g., $S_{(123)}$ corresponds to a hidden state at time $t+3$ obtained by performing action 1 at time t, 2 at time $t+1$ and 3 at time $t+2$. Using this notation, Figure 10 depicts all the possible policies up to two time steps in the future and the associated hidden states. Importantly, Figure 10 shows that the number of policies grows exponentially with the number of time steps for which the agent tries to plan. Therefore, the definition of the prior over the policies, i.e., $P(\pi|\gamma) = \sigma(-\gamma \boldsymbol{G})$, exhibits an exponential space and time complexity class because the agent needs to store and compute the $|U|^T$ parameters of $P(\pi|\gamma)$, where T is the time-horizon.

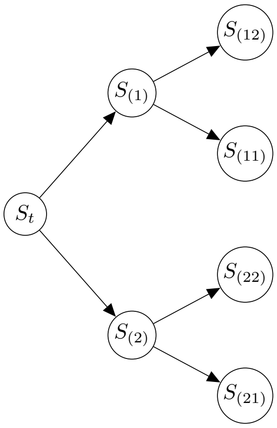


Figure 10: Illustration of all possible policies up to two time steps in the future when $|U| = 2$. The state at the current time step is denoted by S_t. Additionally, each branch of the tree corresponds to a possible policy, and each node S_I is indexed by a multi-index (e.g. $I = (12)$) representing the sequence of actions that led to this state. This should make it clear that for one time step in the future, there are $|U|$ possible policies, after two time steps there are $|U|$ times more policies, and so on until the time-horizon T where there are a total of $|U|^T$ possible policies, i.e., the number of possible policies grows exponentially with the number of time steps for which the agent tries to plan.

To show that this exponential explosion is not only a theoretical problem and also appears in practice, we modified the *DEMO_MDP_maze.m* of the SPM[1] package in two ways. First, we allowed the agent to plan N time steps in the future as follows:

```
% V = Allowable policies (N moves into the future), nu = number of actions
V = [];
for i1 = 1:nu, i2 = 1:nu, .., iN = 1:nu
      V(:,end + 1) = [i1;i2; .. ;iN];
end
```

Second, we used the “tic” and “toc” functions provided by Matlab to track the execution time required to execute the function “*spm_maze_search*” where the parameters of the model are assumed to be known already by the agent:

```
tic % Start a timer used to evaluate the execution time of spm_maze_search
MDP = spm_maze_search(mdp,8,START,END,128,1);
toc % Display the time elapsed since the last call to tic
```

[1] Statistical parametric mapping (SPM) is a software package created by the Wellcome Department of Imaging Neuroscience at University College London, which was initially developed to carry out statistical analyses of functional neuroimaging data. Today, SPM also contains MatLab simulations implementing active inference agents (among other things), c.f. https://www.fil.ion.ucl.ac.uk/spm/.

Figure 11 presents the results of our simulations for N from 2 to 6. Under a logarithmic scale on the time axis, the experimental results show that the graph is almost a perfect line, which provides empirical evidence for an exponential time explosion. Note that the simulation for $N = 7$ crashed after trying to allocate an array of 9.5GB (space explosion). In section 5, we presented an approach proposed to deal with the exponential complexity class that arises during planing, yet is fundamentally similiar to active inference. This effectively means that our paper shows how standard active inference can be made more efficient and scale to longer time horizons.

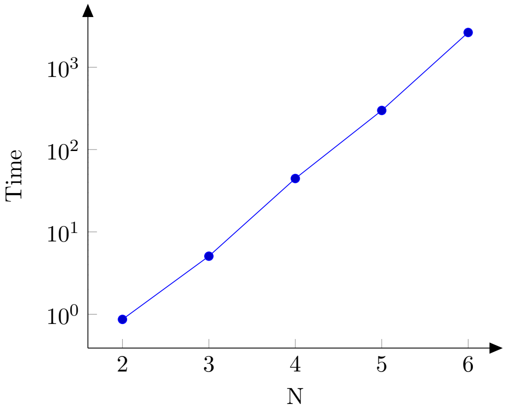


Figure 11: This figure shows the time required to execute the function "*spm_maze_search*" when the agent is allowed to plan N time steps in the future (for N from 2 to 6). A logarithmic scale is used on the time axis.