frontiers in Robotics and AI

ORIGINAL RESEARCH
published: 28 March 2019
doi: 10.3389/frobt.2019.00020

Simulating Active Inference Processes by Message Passing

Thijs W. van de Laar[1]* and *Bert de Vries*[1,2]

[1] Department of Electrical Engineering, Eindhoven University of Technology, Eindhoven, Netherlands, [2] GN Hearing Benelux BV, Eindhoven, Netherlands

The free energy principle (FEP) offers a variational calculus-based description for how biological agents persevere through interactions with their environment. Active inference (AI) is a corollary of the FEP, which states that biological agents act to fulfill prior beliefs about preferred future observations (target priors). Purposeful behavior then results from variational free energy minimization with respect to a generative model of the environment with included target priors. However, manual derivations for free energy minimizing algorithms on custom dynamic models can become tedious and error-prone. While probabilistic programming (PP) techniques enable automatic derivation of inference algorithms on free-form models, full automation of AI requires specialized tools for inference on dynamic models, together with the description of an experimental protocol that governs the interaction between the agent and its simulated environment. The contributions of the present paper are two-fold. Firstly, we illustrate how AI can be automated with the use of ForneyLab, a recent PP toolbox that specializes in variational inference on flexibly definable dynamic models. More specifically, we describe AI agents in a dynamic environment as probabilistic state space models (SSM) and perform inference for perception and control in these agents by message passing on a factor graph representation of the SSM. Secondly, we propose a formal experimental protocol for simulated AI. We exemplify how this protocol leads to goal-directed behavior for flexibly definable AI agents in two classical RL examples, namely the Bayesian thermostat and the mountain car parking problems.

Keywords: active inference, free-energy principle, message passing, state-space models, Forney-style factor graphs

OPEN ACCESS

Edited by:
Fabrizio Riguzzi,
University of Ferrara, Italy

Reviewed by:
Biswa Sengupta,
Independent Researcher, London, United Kingdom
Dimitri Ognibene,
University of Essex, United Kingdom

****Correspondence:***
Thijs W. van de Laar
t.w.v.d.laar@tue.nl

Specialty section:
This article was submitted to Computational Intelligence, a section of the journal Frontiers in Robotics and AI

Received: *22 November 2018*
Accepted: *05 March 2019*
Published: *28 March 2019*

Citation:
van de Laar TW and de Vries B (2019) Simulating Active Inference Processes by Message Passing. Front. Robot. AI 6:20. doi: 10.3389/frobt.2019.00020

1. INTRODUCTION

The free energy principle (FEP) offers an ambitious theory for how biological agents perceive and interact with their environment (Friston, 2009, 2010). The FEP postulates that in order for an agent to exist (and persist) under time-varying environmental conditions, it must minimize a free energy functional under the agent's internal ("generative") model for environmental observations (Friston et al., 2006).

Active inference, which is a corollary of the free energy principle, claims that natural agents act to fulfill prior beliefs about preferred observations (Friston, 2010). These prior beliefs about future observations are part of the agent's internal model specification, and free energy minimization thus ensures that the agent avoids surprising states.

Currently, the derivation of active inference algorithms on free-form dynamic models still requires manual work. Automation of active inference processes might enable practitioners to build more effective, flexible and scalable agents (de Vries and Friston, 2017). In addition, automated execution of active inference processes also requires the definition of a formal experimental protocol that governs the interaction between the agent and its environment.

The derivation of a free energy minimizing algorithm can be automated through the use of probabilistic programming (PP) techniques (Tran et al., 2016; Carpenter et al., 2017; Minka et al., 2018). While most PP toolboxes offer free-form modeling tools and automated derivation of flexible inference algorithms, their generality often comes at the cost of increased computational load. In contrast, dynamic models incorporate model-specific structure that can be exploited to increase algorithm performance. Here, message passing on a factor graph description of the generative model is especially suited for inference in flexibly definable dynamic models (Loeliger et al., 2007; Cox et al., 2019).

The current paper details an experimental protocol and simulation environment for the automated derivation and execution of online active inference processes in a dynamic context. Crucially, we illustrate how the message passing approach and proposed experimental protocol cooperate to automate the execution of structured active inference algorithms on flexibly definable generative dynamic models. We address the following questions:

1. How can online active inference processes be operationally described by an experimental protocol?
2. How can active inference processes be automatically derived within the given protocol?

With respect to the first issue, we describe a protocol that formally captures the interactions between an (active inference) agent and its environment. The protocol supports online simulations under situated conditions.

Concerning the second issue, the current paper provides a full message-passing based account of active inference, formulated in a Forney-style factor graph (FFG) representation of the internal model (Forney, 2001). The FFG formulation supports flexible model definitions and *automated* (active) inference execution by message passing-based free energy minimization. Crucially, this automation absolves the need for manual derivations of variational calculus problems and in principle scales up to complex hierarchical and interdependent models, making the approach suitable for industrial-sized applications.

The paper is structured as follows. Sections 2 and 3 offer a short technical rehearsal to active inference and FFGs, respectively. The experimental protocol is detailed in section 4. In section 5 we test our proposed protocol by simulating two classical active inference applications, namely the Bayesian thermostat and the mountain car. These simulations were executed with ForneyLab, which is a freely available toolbox for automated free energy minimization in FFGs that we have developed in our research lab (Cox et al., 2019). Finally, we discuss related work in section 6 and conclude in section 7.

2. ACTIVE INFERENCE

Friston (2013) considers the consequences of being alive for the internal informational mechanics of natural agents. This analysis leads to the conclusion that natural agents appear to exchange information with their environment so as to maximize Bayesian evidence (i.e., minimize free energy) for an internal model of sensory data. We consider the system of interacting agent and environment as drawn in **Figure 1**. Both the agent and environment are considered dynamical systems with hidden state dynamics. The environment executes a process $(y_t, z_t) = R_t(z_{t-1}, a_t)$, where a_t is an action, z_t a latent state vector and y_t is the output signal. The agent holds a generative probabilistic model $p_t(x, s, u)$ for the environmental process, where x, s, and u are sequences of observations, internal states and controls. The action at time t in the environmental process is represented in the agent's model by control variable u_t and environmental output (y_t) by sensory variable ("observation") x_t. The agent has a single goal, namely minimizing free energy, which roughly corresponds to minimizing (precision-weighted) prediction errors for its sensory inputs x (Rao and Ballard, 1999; Friston and Kiebel, 2009; Huang and Rao, 2011). Minimizing free energy leads to posterior beliefs over both the states s (which affects predictions and thereby prediction errors) and controls, which are observed by the environment as actions that lead to changes in the sensory inputs. These changes in sensory inputs may also affect prediction errors. Thus, inference for the states and controls both lead to free energy minimization. Technically, in order for a natural agent to persevere, it must be both physically and statistically separated from its environment. This statistical skin is called a *Markov Blanket*, which comprises the sensory and action variables. Sensory variables are affected by internal states of the environment but do not directly affect internal environmental

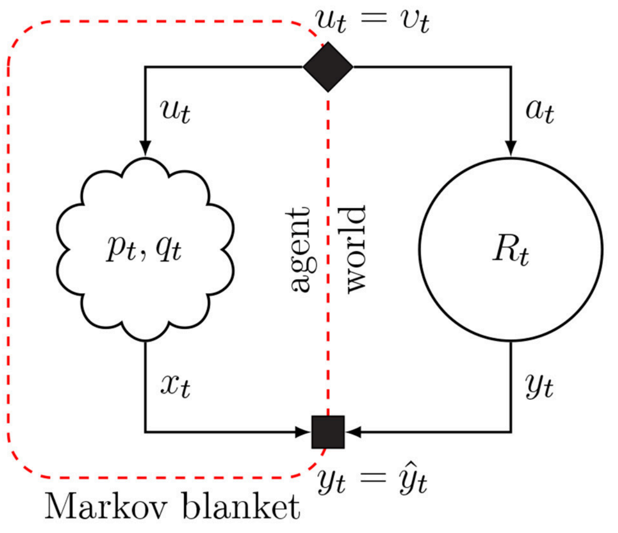

FIGURE 1 | The Markov blanket forms a statistical separation between an agent and its environment. Arrows indicate the "generative" direction. Conditioning on the Markov blanket renders information about the internal states of the agent independent from information about hidden state of the environment. Blanket states for the agent are the variables for controls u_t and observations x_t. The solid diamond node in the Markov blanket links posterior beliefs over the agent's control variables to environmental actions, and the solid square node in the Markov blanket passes environmental outcomes to agent observations.

states. Similarly, actions are affected by internal states of the agent but do not directly affect internal states of the agent.

Because active inference reasons from observations toward controls, the inference process requires the definition of an "inverse" probabilistic model which is sometimes called the *recognition model* q_t. Full Bayesian inversion of p_t is in general intractable, so the agent resorts to approximating Bayesian inference by minimizing a variational free energy functional, defined by:

$$F[q_t] = \int_{q_t} q_t(s,u) \log \frac{q_t(s,u)}{p_t(x,s,u)} \tag{1a}$$

$$= \underbrace{-\log p_t(x)}_{\text{surprise}} + \underbrace{\int_{q_t} q_t(s,u) \log \frac{q_t(s,u)}{p_t(s,u|x)}}_{\text{posterior divergence}} \tag{1b}$$

Minimizing Equation (1) renders the free energy an upper bound on surprise (negative log-evidence), while simultaneously approximating the (generally unavailable) true posterior $p_t(s,u|x)$ with the recognition model q_t. In order to render this optimization process tractable, the recognition model is often factorized (Attias, 1999), where a fully factorized recognition model is referred to as the "mean-field" assumption.

In order to equip the agent with a sense of "goal-directedness" or "purpose," the internal model extends over future states and incorporates counter-factual beliefs about desired future outcomes, also referred to as *target priors* (Parr and Friston, 2018). These target priors lead to high surprisal for observations that are unlikely under the agent's preferences. Free energy minimization then produces (approximate) posterior beliefs over controls that are believed (by the agent) to avoid these undesired (surprising) observations. In the current paper we set the target priors ourselves, but more generally these priors might be set by contextual processes such as other agents or higher-level temporal layers. These ideas are further discussed in section 7.

Previous accounts of active inference introduce an expected free energy that steers goal-directed behavior via a prior over control (Friston et al., 2015; Friston K.J. et al., 2017). Instead, we will employ the internal model formulation by Parr and Friston (2018), which explicitly includes counter-factual prior beliefs over future observations in order to steer behavior. This leads to a unified model specification that allows us to optimize a single free energy functional (see also de Vries and Friston, 2017). Minimizing this functional by message passing simultaneously captures inference over current states (perception) as well as future controls (action/policy planning).

Practical models for active inference that are suitable for industrial application may be complex, layered and embedded in a volatile context. Manual derivation of free energy minimizing algorithms for active inference algorithms will then become prohibitively tedious and error-prone. A graphical representation of the internal model will aid with visualization of complex models and allows for automated derivation of message passing algorithms. The next section introduces Forney-style factor graphs as a graphical framework for automatic derivation of active inference algorithms.

3. INFERENCE BY MESSAGE PASSING ON FORNEY-STYLE FACTOR GRAPHS

A Forney-style factor graph (FFG), also known as a "normal" factor graph, offers a graphical description of a factorized function (Forney, 2001). Excellent and detailed introductions to FFGs are available in (Loeliger, 2004; Korl, 2005). While related graphical formalisms such as Bayesian networks, Markov random fields and bipartite factor graphs offer essentially equivalent formulations (Forney, 2001; Loeliger, 2004), the FFG formalism is especially suited for representing dynamical models (Loeliger et al., 2007). Specifically, the FFG representation requires only a single node and message type, while retaining the explicit representation of variable relations through factor nodes.

As an example factorization, in this section we consider the function of Equation (2), which splits into four factors:

$$f(x_1,x_2,x_3,x_4) = f_a(x_1)\, f_b(x_1,x_2,x_4)\, f_c(x_2,x_3)\, f_d(x_4)\,. \tag{2}$$

In this paper, we assume that the function f is a probability distribution. The FFG for this factorization is drawn in **Figure 2** (middle), together with the equivalent bipartite factor graph representation (left) for comparison. In an FFG, variables

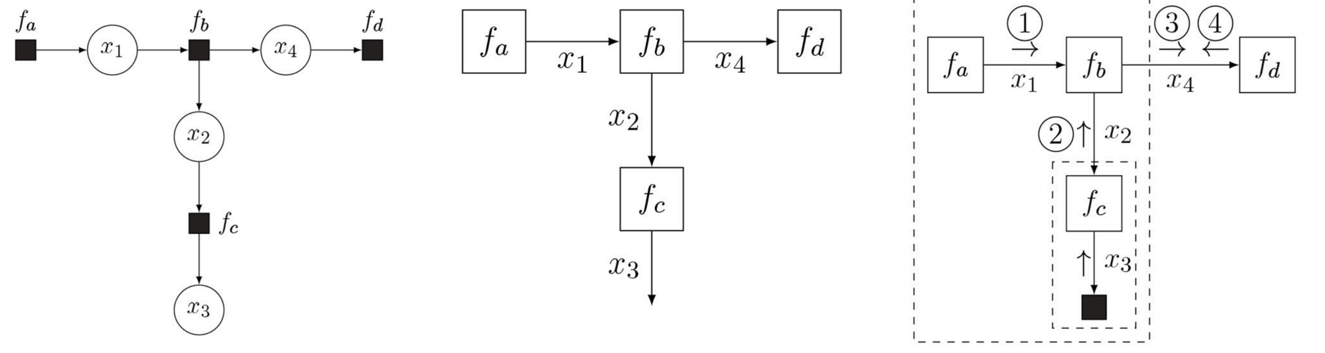

FIGURE 2 | Bipartite factor graph **(left)** and Forney-style factor graph (FFG) **(middle)** for the model of Equation (2), together with the message passing schedule for computing (Equation 3b) **(right)**. In a bipartite factor graph, solid nodes represent factors and round nodes represent variables. In an FFG, edges represent variables and all nodes represent factors, where solid nodes indicate observations.

correspond to edges and factors are represented by nodes. A node is connected to an edge only if the variable of the edge is an argument in the factor. For instance, node f_c connects to edges x_2 and x_3 since $f_c = f_c(x_2, x_3)$.

Now assume that we observe $x_3 = \hat{x}_3$ (technically: x_3 is a variable that takes on value $\hat{x}_3$) and are interested in computing a posterior distribution for x_4. The observation of x_3 then imposes an additional constraint $\delta(x_3 - \hat{x}_3)$ on the model, which clamps this variable to its observed value. Following (Reller, 2012), we indicate observations by a small solid node, see **Figure 2** (right).

The integral for computing the posterior distribution is shown in Equation (3a). A direct approach to solving this integral might be tedious and error-prone. Conveniently, the distributive law allows us to pull unaffected integrands out of integrals, thus unpacking the total integral into a product of sub-integrals, each of which can be interpreted as a message over an edge of the factor graph, cf. **Figure 2** (right) and Equation (3b):

$$f(x_4|x_3 = \hat{x}_3) \propto \iiint f(x_1, x_2, x_3, x_4)\, \delta(x_3 - \hat{x}_3)\, \mathrm{d}x_1\, \mathrm{d}x_2\, \mathrm{d}x_3 \tag{3a}$$

$$= \underbrace{\iint \overbrace{f_a(x_1)}^{\textcircled{1}} \overbrace{\int f_c(x_2, x_3)\, \delta(x_3 - \hat{x}_3)\, \mathrm{d}x_3}^{\textcircled{2}}\, f_b(x_1, x_2, x_4)\, \mathrm{d}x_1\, \mathrm{d}x_2}_{\textcircled{3}} \times \underbrace{f_d(x_4)}_{\textcircled{4}} \tag{3b}$$

$$= \overrightarrow{\mu}_{\textcircled{3}}(x_4)\, \overleftarrow{\mu}_{\textcircled{4}}(x_4). \tag{3c}$$

Consecutive computation of these messages then leads to the solution of Equation (3c), where the (unnormalized) posterior function results from the product of the two colliding messages $\textcircled{3}$ and $\textcircled{4}$.

For efficiency reasons the message normalizing constants are often ignored, and messages are represented by a probability distribution with corresponding sufficient statistics. The message can then be interpreted as an information summary for the variable of the corresponding edge. For instance, message $\textcircled{3}$ in **Figure 2** (right) represents the probability distribution for x_4 given the information in the large box at left-hand side of message $\textcircled{3}$. While an FFG is principally an undirected graph, we often draw arrows on the edges in order to anchor the message direction. A forward message $\overrightarrow{\mu}$ aligns with the edge arrow, and a backward message $\overleftarrow{\mu}$ aligns against the edge direction.

Crucially, the message passing approach to inference allows to reuse pre-derived solutions to specific message updates for elementary factors across multiple models. Implementing these solutions in a look-up table allows us to automate the derivation and execution of message passing algorithms. Well-known algorithms such as (loopy) belief propagation (Forney, 2001), variational message passing (Dauwels, 2007), expectation maximization (Dauwels et al., 2005), and expectation propagation (Cox, 2018) have all been formulated as message passing procedures on an FFG.

3.1. Example: Equality Node

In order to exemplify the derivation of a reusable message update rule for an elementary factor, we consider the equality factor (see also Korl, 2005; Cox et al., 2019), defined as

$$f_=(x, y, z) = \delta(z - x)\, \delta(z - y)\,. \tag{4}$$

In contrast to a variable in a bipartite graph (**Figure 2**, left), a variable in an FFG can connect to at most two factors. The equality factor resolves this situation by constraining the information about three variables to be equal (Equation 4). Edges constrained by equality factors can then effectively be regarded as a single variable that is shared among connected factors.

An update for the equality node is computed by Equation (5b), for which the schedule is graphically represented by **Figure 3** (left). It can be seen that this schedule fuses information from two branches into a single message, much like Bayes rule does. Indeed, in FFGs, equality nodes are often used to connect prior information about a variable with likelihood-based information about that variable.

$$\overrightarrow{\mu}_{\textcircled{3}}(z) = \iint \overrightarrow{\mu}_{\textcircled{1}}(x)\, \overrightarrow{\mu}_{\textcircled{2}}(y)\, f_=(x, y, z)\, \mathrm{d}x\, \mathrm{d}y \tag{5a}$$

$$= \overrightarrow{\mu}_{\textcircled{1}}(z)\, \overrightarrow{\mu}_{\textcircled{2}}(z)\,. \tag{5b}$$

3.2. Dealing With Nonlinear Factors

The modularity of the message passing approach allows to make local approximations. Through local linearization, we can pass messages through nodes that encode nonlinear constraints. Here

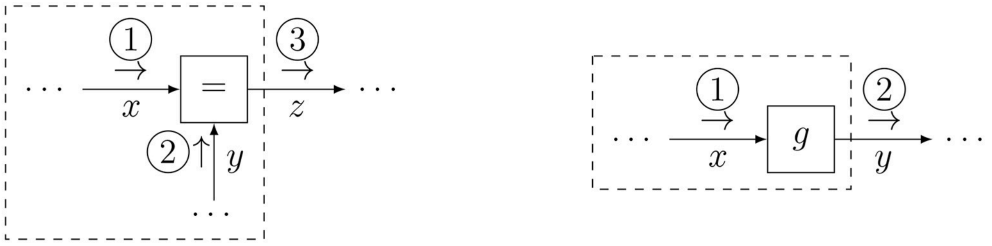

FIGURE 3 | Message updates for the equality node **(left)** and the nonlinear node **(right)**.

we consider the factor of Equation (6), where $g(x)$ is a nonlinear differentiable function:

$$f_g(x, y) = \delta(y - g(x)) \,. \tag{6}$$

The forward message is expressed as Equation (7a) and drawn in **Figure 3** (right).

$$\overrightarrow{\mu}_{\textcircled{2}}(y) = \int \overrightarrow{\mu}_{\textcircled{1}}(x) f_g(x, y)\, \mathrm{d}x \tag{7a}$$

$$= \overrightarrow{\mu}_{\textcircled{1}}(g^{-1}(y)) \,. \tag{7b}$$

When the incoming message $\textcircled{1}$ is in the exponential family, it can be conveniently parameterized by its sufficient statistics. However, because of the nonlinear transformation, the outgoing message is no longer guaranteed to be a member of the exponential family, which may lead to intractable updates. Therefore, we trade some accuracy for computability by making a local linear approximation to the node function. We choose the approximation point $\hat{x}$ as the mean of the incoming message, and expand g locally as

$$\tilde{g}(x) = g(\hat{x}) + J_g(\hat{x})(x - \hat{x}) \,, \tag{8}$$

where $J_g(\hat{x})$ is the Jacobi matrix at the approximation point (or the first derivative in the scalar case). By substituting Equation (8) in Equation (6), we can obtain approximate but tractable message updates. This local linearization strategy for nonlinear factors will be used in section 5.2.

4. ONLINE ACTIVE INFERENCE IN FACTOR GRAPHS

In section 2 we mentioned the agent's internal model p_t, which expresses the agent's prior beliefs about how the environmental process generates observations from actions. In the current section we propose a simulation protocol for online active inference execution by the agent.

4.1. Model Specification

We consider an agent with a state-space model (SSM) factorization for its internal model, given by

$$p_t(x, s, u) \propto p(s_{t-1}) \prod_{k=t}^{t+T} \underbrace{p(x_k \mid s_k)}_{\text{observation}} \underbrace{p(s_k \mid s_{k-1}, u_k)}_{\text{state transition}} \underbrace{p(u_k)}_{\text{control}} \underbrace{p'(x_k)}_{\text{target}} \,. \tag{9}$$

where x, s, and u are sequences with ranges that are implicitly given by the model specification at the right-hand side. Note how the model of Equation (9) differs slightly from a standard SSM factorization (Koller and Friedman, 2009), because it includes additional "target" priors $p'(x_k)$ over desired future outcomes. A factor graph representation of Equation (9) is shown in **Figure 4**. Also note that the probability distribution for the internal model has a subscript t to indicate that the model is time-varying. We will consider these aspects in more detail below.

Also note that at time step t, the agent has assumptions about how the environment will evolve over the next $T + 1$ time steps since we can run this model forward and generate observations x_k for $k = t, t + 1, \ldots, t + T$. The horizon T is determined by the information content of the target priors $p'(x_k)$. These target priors are generally set by states of contextual processes, i.e., not by this agent but rather by other agents (or higher level processes) that encode unsurprising future outcomes for this agent. In order to distinguish the predictive model for observations $p(x_k|s_k)$ from the context-based target prior for observations $p'(x_k)$, we label the latter factor with a prime.

We refer to a sequence of future controls $u = (u_t, u_{t+1}, \ldots, u_{t+T})$ as a *policy*. Through inference, the posterior over policies becomes dependent on the hidden state sequence s. Prior to inference however, the model requires the definition of a prior belief over policies that constrains attainable control states. In a more general formulation of the internal model, we would write a prior over policies $p(u) = p(u_t, u_{t+1}, \ldots, u_{t+T})$. Here, for simplicity, we assume $p(u) = \prod_{k=t}^{t+T} p(u_k)$.

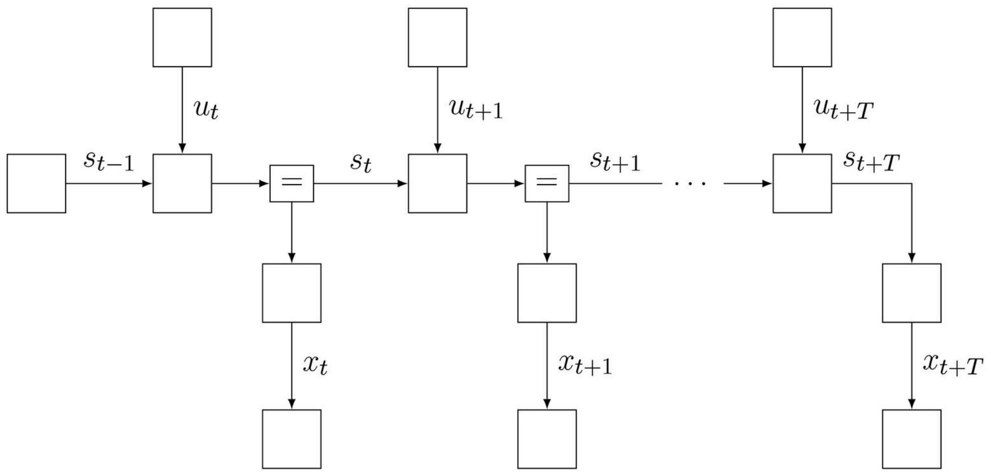

FIGURE 4 | The Forney-style factor graph for the agent's generative model (Equation 9).

Next to the internal model, we assume that the agent has access to a variational distribution, also known as the *recognition* distribution,

$$q_t(x, s, u) \tag{10}$$

that encodes the agent's posterior beliefs about all latent states. Because the future is by definition unobserved, the recognition distribution includes the future observation variables as well. The distinction between the agent's prior ("generative") beliefs $p_t(x, s, u)$ and posterior ("recognition") beliefs $q_t(x, s, u)$ will be finessed below. At the start of the simulation (say at $t = 1$), we will set q_1 to an uninformative distribution. As time progresses, observations become available and subsequent inference iterations will tune q_t to more informed distributions. Often, but not necessarily so, the recognition distribution is assumed to be fully factorized (this is the *mean-field* assumption). In the present article we assume a structured factorization for the recognition distribution that is solely induced by the internal model factorization (Bishop, 2006). In general, we may write

$$q_t(x, s, u) = q(x, s|u)\, q(u)\,.$$

Since actions onto the environment are real-valued interventions, we will generally enforce a deterministic posterior belief over policies, i.e.,

$$q(u_t, \ldots, u_{t+T}) = \prod_{k=t}^{t+T} \delta(u_k - \upsilon_k)\,, \tag{11}$$

where υ_k (upsilon) are parameters that are to be determined in the free-energy minimization process (see section 4.3). In order words, while the prior belief over policies $p(u)$ may contain uncertainties, we will fix the posterior belief over policies $q(u)$ on a single concrete sequence.

As time progresses, at each time step, the agent interacts with its environment through exchanging actions and observations, followed by processing the observations. Technically, everything that the agents does can be described as updates to its internal and recognition models. We distinguish three phases per time step: (1) act-execute-observe, (2) infer (process observations), and (3) slide (move one time slice ahead), see **Algorithm 1**. Next, we consider each step in more detail.

4.2. The "Act-Execute-Observe" Step

Since we assumed that the posterior beliefs over control states were constrained by $\delta(u_k - \upsilon_k)$, we will select the *action* at time step t as

$$a_t = \upsilon_t\,. \tag{12}$$

Algorithm 1 The online active inference algorithm

```
Require: $p_1(x, s, u)$ and $q_1(x, s, u)$ //model specification
1: for $t = 1$ to $\infty$ do
2:     Act-Execute-Observe // update to $p_t(x, s, u)$
3:     Infer // update to $q_t(x, s, u)$
4:     Slide // move one time slice ahead
5: end for
```

Technically, in a factor graph context, we will let the agent be the owner of the control state u_t and the environment is the owner of action variable a_t. The agent and environment are coupled at the agent's Markov blanket by an interface factor $\delta(u_t - a_t)$. Message passing from the agent to the environment will now pass the agent's posterior belief $q(u_t) = \delta(u_t - \upsilon_t)$ over control state u_t to the action, leading to $a_t = \upsilon_t$. Next, this action is imposed onto the environmental process R_t that generates outputs y_t by

$$(y_t, z_t) = R_t(z_{t-1}, a_t)\,, \tag{13}$$

where z_t refers to the dynamic states of the environmental process. Here, we call environmental processing the *execution* phase. In similar fashion to the action-interface, if we let y_t refer to the output *variable* and $\hat{y}_t$ is the *value* of the environmental output at time t, then an observation-interface node $\delta(x_t - y_t)$ at the agent's Markov blanket leads to an incoming message $\delta(x_t - \hat{y}_t)$ from observation variable x_t. In a real-world setting, the agent does not know the model R_t nor the environmental states, but still gets to observe the environmental output. Acting onto the environment and observing the consequences can technically be processed by the agent through updating its internal model to

$$p_t(x, s, u) := p_t(x, s, u)\, \underbrace{\delta(u_t - \upsilon_t)\, \delta(x_t - \hat{y}_t)}_{\text{action and observation}}\,. \tag{14}$$

The factor graph of the updated internal model is shown in **Figure 5**. We use small filled diamond nodes to indicate causal interventions (actions) and small square nodes for observation variables. Note that when we multiply the model with a delta distribution $\delta(u_t - \upsilon_t)$, we technically overwrite the prior $p(u_t)$ and the open square for $p(u_t)$ is replaced by a black diamond smaller square to indicate the intervention. Similarly, the target prior $p'(x_t)$ is omitted because the observation $\delta(x_t - \hat{y}_t)$ renders it conditionally independent from the rest of the model.

4.3. The "Infer" Step

The internal model has changed as a result of acting and observing. In the infer step, we process the consequences of this change for the model's latent variables. Technically, we update the posterior from q_{t-1} to q_t by free energy minimization. Generally, the recognition distribution will be parameterized by sufficient statistics μ, so we can write $q_t(x, s, u) = q(x, s, u|\mu_t)$ and free energy minimization amounts to finding μ_t by

$$\mu_t = \arg\min_{\mu} \underbrace{\int q(x, s, u|\mu) \log \frac{q(x, s, u|\mu)}{p_t(x, s, u)}\, \mathrm{d}x\, \mathrm{d}s\, \mathrm{d}u}_{\text{free energy } F_t(\mu) \text{ at time step } t} \tag{15}$$

This minimization procedure can be executed through variational message passing (VMP) in the factor graph for the updated internal model $p_t(x, s, u)$. The inference process is visualized in **Figure 6**. VMP involves iteratively updating single (or a few) components of μ while holding the other components fixed. In this procedure, it will be very useful to set the initial value μ_t^{init} to μ_{t-1}. With this initialization, in most cases, the number of VMP iterations to convergence per time step will be quite low.

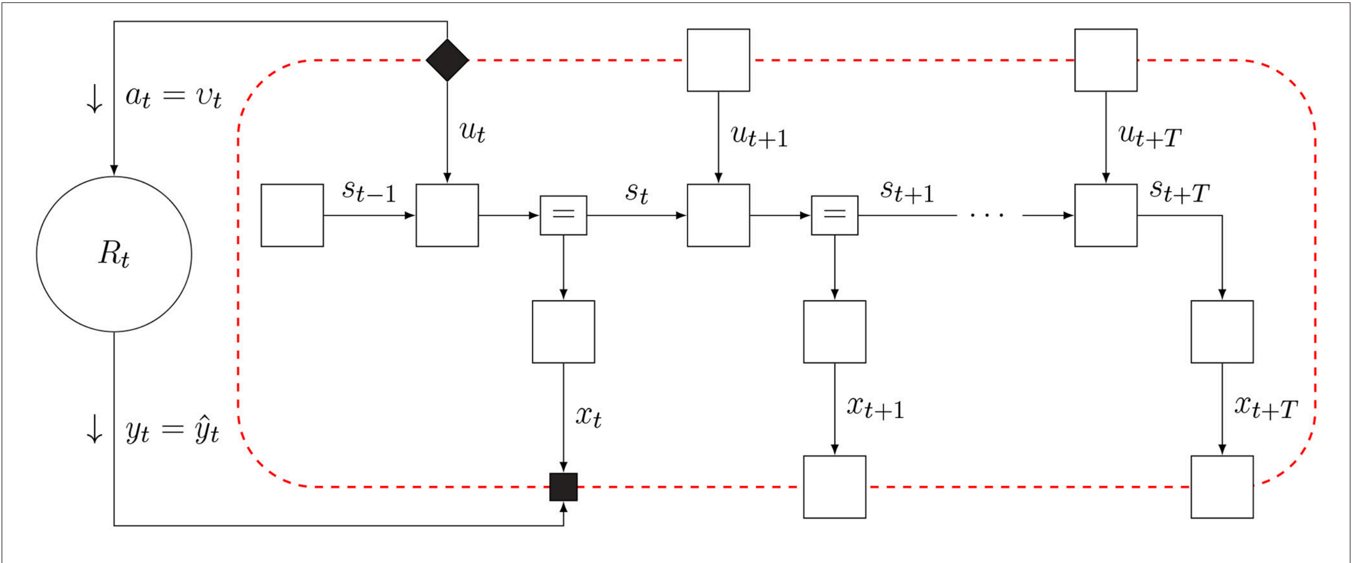

FIGURE 5 | The FFG for the act-execute-observe phase.

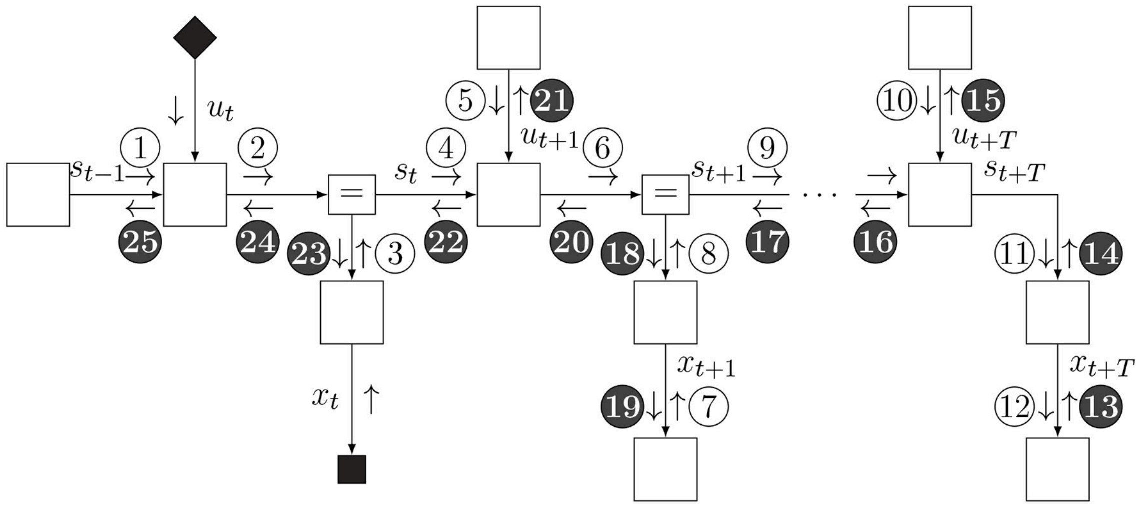

FIGURE 6 | Schedule for infer phase. Light messages constitute a forward (filtering) pass, and dark messages constitute a backward (smoothing) pass.

4.4. The "Slide" Step

The slide step implements a time step increment and is best understood by looking at the factor graph in **Figure 7**. Essentially, we eliminate (marginalize out) the first time slice of the internal and recognition models and add a time slice to the horizon. Then we fix the indexing back to p_t.

Formally, marginalization of the first slice and adding a slice at the horizon leads to the following update of the agent's internal model:

$$p_{t+1}(x, s, u) \propto \underbrace{\int \overbrace{p(s_{t-1})\, p(s_t|s_{t-1}, u_t)\, p(x_t|s_t)}^{\text{first slice}}\ \overbrace{\delta(u_t - a_t)\, \delta(x_t - y_t)}^{\text{action and observation}}\ \mathrm{d}s_{t-1}\mathrm{d}u_t\mathrm{d}x_t}_{\text{close box (marginalization) for first slice, yielding new prior } p(s_t)} \cdot \underbrace{\left(\prod_{k=t+1}^{t+T} p(x_k|s_k)\, p(s_k|s_{k-1}, u_k)\, p(u_k)\, p'(x_k)\right)}_{\text{unaltered mid-section slices}} \cdot \underbrace{p(x_{t+T+1}|s_{t+T+1})\, p(s_{t+T+1}|s_{t+T}, u_{t+T+1})\, p(u_{t+T+1})\, p'(x_{t+T+1})}_{\text{add slice at horizon}} \qquad (16)$$

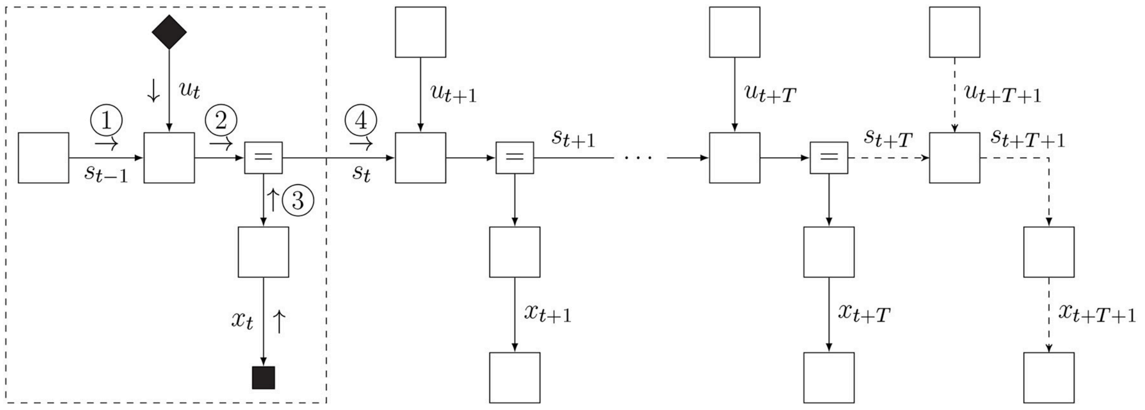

FIGURE 7 | Schedule and FFG extension for slide phase.

Here, we have processed the action and observation together with sliding a time step in one update. The marginalization of x_t, s_{t-1}, and u_t in the first slice of the graph can be executed by message passing. In practice, we can simply re-use message 4 from **Figure 6** (after re-normalization) as the new state prior $p(s_t)$. In order to maintain a horizon of T time steps, we add one time slice of the state space model to the horizon (at time step $t + T + 1$). Again, it is assumed that a contextual agent feeds the target prior $p'(x_{t+T+1})$.

Adding and deleting slices at each time step also provides room for changing the model structure of the internal model on the basis of current contextual information. For instance, the observation model $p(x_{t+T+1}|s_{t+T+1})$ does not need to be the same as $p(x_{t+T}|s_{t+T})$. The decision for the structure of $p(x_{t+T+1}|s_{t+T+1})$ can be postponed until time step t.

Because the recognition model is simply parameterized by its sufficient statistics $[q_t(x, s, u) = q(x, s, u|\mu_t)]$, we do not require an explicit slide for the recognition model. In the next "infer" step, the recognition model is simply initialized with the present statistics ($\mu_t^{init} = \mu_{t-1}$) and updated to the new posterior statistics, without the need for redefining the recognition model.

Once the internal model has slided forward by one slice, we increment the time step index by

$$t := t + 1 \tag{17}$$

so as to obtain a generative model $p_t(x, s, u)$ again (see Equation 9), but now for an increased time step. The Slide phase is followed by repeating the loop, starting with an Act-Execute-Observe step.

In summary, online active inference by a dynamic agent proceeds according to updating both its internal and recognition distributions at each time step according to Equations 9, 11, 15, 16, and 17. Actions and observations are technically processed by appending delta factors to the agent's internal model. The effects of these internal model changes on the latent variables in the model are inferred through free energy minimization, which leads to an updated recognition distribution. Next, the model slides forward one time slice and starts a new Act-Execute-Observe step.

5. SIMULATIONS

In order to illustrate the operationability of the active inference protocol, we simulate two classic active inference problems, namely the Bayesian thermostat (Friston et al., 2012; Buckley et al., 2017) and the mountain car (Friston et al., 2009; Sutton and Barto, 2018; Ueltzhöffer, 2018). We use ForneyLab (version 0.9.1) as a tool for automated derivation of message passing algorithms (Cox et al., 2019). ForneyLab (available from https://github.com/biaslab/ForneyLab.jl) supports flexible specifications of factorized probabilistic dynamic models (van de Laar et al., 2018) and generates high-performance inference algorithms on these models (Cox et al., 2019).

ForneyLab is written in Julia, a high-level programming language for numerical computing. Julia combines an accessible MATLAB-like syntax with C-like performance (Bezanson et al., 2017). Julia's excellent meta-programming capabilities allow ForneyLab to define an intuitive domain-specific model specification syntax, and to automatically compile message passing schedules directly to executable Julia code. Furthermore, Julia's powerful multiple dispatch functionality enables efficient scheduling and algorithm execution.

5.1. The Bayesian Thermostat

In this section we simulate an agent that can relocate itself in a temperature gradient field. The agent aims to position itself at a desired temperature relative to a heat source. Our simulation setup is an adaptation of the setup described by Buckley et al. (2017).

5.1.1. Environmental Process Specification

First we specify the environmental process. The temperature $\mathcal{T}$ as a function of position z follows the profile

$$\mathcal{T}(z) = \frac{\mathcal{T}_0}{z^2 + 1}, \tag{18}$$

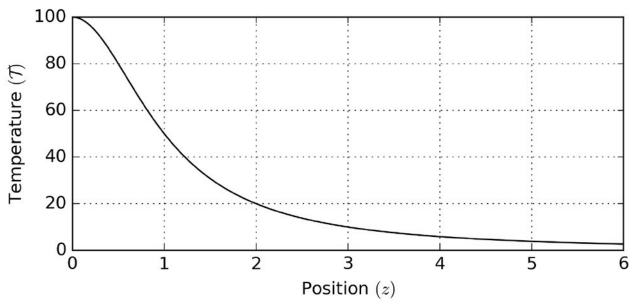

FIGURE 8 | Environmental temperature profile, showing environmental temperature as a function of position.

where $\mathcal{T}_0 = 100$ represents the temperature at the location of the heat source, see **Figure 8**.

The environmental process is steered by actions a_t that reflect the velocity of the agent. The output of the environmental process is the observed temperature by the agent. We assume that the agent observes a noisy temperature, leading to the following environmental process equations:

$$z_t = z_{t-1} + a_t \tag{19a}$$

$$y_t \sim \mathcal{N}\big(\mathcal{T}(z_t), \vartheta\big)\ . \tag{19b}$$

In our simulation, we used initial position $z_0 = 2$ and observation noise variance $\vartheta = 10^{-2}$.

5.1.2. Internal Model Specification

The agent's internal model follows the factorization of Equation (9). The goal prior encodes a belief about a preferred temperature $x_\oplus = 4$, as

$$p'(x_k) = \mathcal{N}\big(x_k | x_\oplus, 10^{-2}\big)\ . \tag{20}$$

Furthermore, we endow the agent with an accurate model of system dynamics

$$p(s_k | s_{k-1}, u_k) = \mathcal{N}\big(s_k | s_{k-1} + u_k, 10^{-2}\big)\ . \tag{21}$$

However, in order to challenge the agent, we hamper the observation model. Instead of the actual temperature profile (Equation 19b), we use

$$p(x_k | s_k) = \mathcal{N}\big(x_k | -s_k, 10^{-2}\big)\ , \tag{22}$$

which simply specifies that the observed temperature decreases with position. The internal model definition is completed by specifying a prior for controls and a vague prior for the initial state:

$$p(u_k) = \mathcal{N}\big(u_k | 0, 10^{-2}\big) \tag{23a}$$

$$p(s_0) = \mathcal{N}\big(s_0 | 0, 10^{12}\big)\ . \tag{23b}$$

Substituting Equations (20–23) in Equation (9) then returns the full internal model definition. One section of the generative model FFG is shown in **Figure 9**. A Julia code snippet[1] for constructing the internal model with ForneyLab is shown in **Figure 10**.

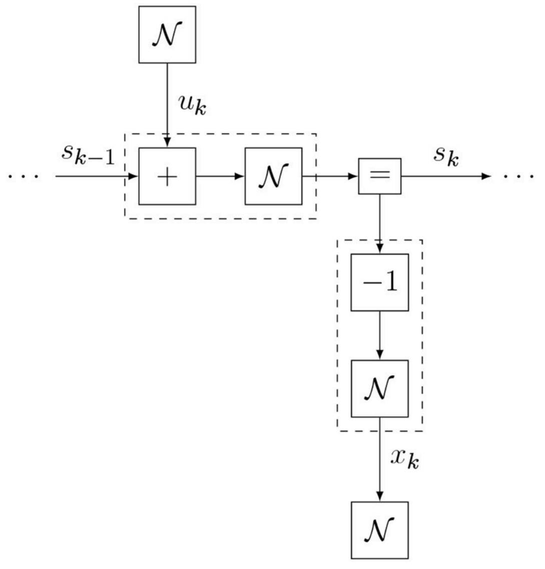

FIGURE 9 | One time section of the Bayesian thermostat model (excluding the state prior).

After having specified the generative model, ForneyLab can be used to automatically generate a message passing algorithm for free energy minimization, see **Figure 11**. ForneyLab generates Julia source code for the message passing algorithm, see **Figure 12** for a snippet of the generated code.

5.1.3. Simulation Results

The agent-environment system was simulated using the experimental protocol as defined in section 4, for 100 time steps and with horizon $T = 2$, with the additional constraint that the agent is only allowed to act after $t = 25$. Execution of the experimental protocol amounts to executing the code of **Figure 13**. Note that this implementation of the experimental protocol ensures that the agent never observes the environmental states of the environment directly, but rather only interacts with the environmental process through the `act`, `execute`, and `observe` functions. This setup emphasizes the idea that the environmental states (z) are not (directly) observable. In principle, this setup is then applicable to a real-world setting where the environmental process is not a simulation.

The graphs of **Figure 14** show the simulation results. It can be seen that after the agent is allowed to act, it quickly moves away from the heat source in order to eventually settle at the desired temperature. Apparently, despite the hampered observation model, the agent still attains the desired goal.

[1]Code for the complete simulation is available at http://biaslab.github.io/materials/ai_simulations.zip.

```
x = Vector{Variable}(undef, T) # Observed states
s = Vector{Variable}(undef, T) # Brain states
u = Vector{Variable}(undef, T) # Control states (policy)

@RV s_t_min   GaussianMeanVariance(placeholder(:m_s_t_min),
                                   placeholder(:v_s_t_min)) # Internal state prior
s_k_min = s_t_min
for k = 1 :T # For present and future timepoints
    @RV u[k]   GaussianMeanVariance(placeholder(:m_u, var_id=:m_u_*k, index=k),
                                    placeholder(:v_u, var_id=:v_u_*k, index=k)) # Control prior
    @RV s[k]   GaussianMeanPrecision(s_k_min + u[k], 1e2) # State transition model
    @RV x[k]   GaussianMeanVariance(-1.0*s[k], 1e-2) # Observation model
    GaussianMeanVariance(x[k],
                         placeholder(:m_x, var_id=:m_x_*k, index=k),
                         placeholder(:v_x, var_id=:v_x_*k, index=k)) # Target prior
    s_k_min = s[k]
end
```

FIGURE 10 | Julia code for building the internal model for the Bayesian thermostat with ForneyLab. In Julia, the prefix "@" indicates a macro. Under the hood, `@RV` constructs the Forney-style factor graph for the specified random variables. The ":" prefix (e.g., `:m_x`) specifies a symbol that may be used for indexing. `placeholder` indicates a placeholder for data that can be passed to the algorithm during inference.

```
algo = sumProductAlgorithm([s; u]) # Generate message passing algorithm code
```

FIGURE 11 | ForneyLab command for automated generation of the message passing algorithm for inference in the Bayesian thermostat simulation. The sum-product algorithm implicitly assumes the recognition model of Equation (11). ForneyLab also supports generating message passing code for alternative constraints on the recognition distribution (see e.g., Cox et al., 2018).

```
function step!(data::Dict, marginals::Dict=Dict(), messages::Vector{Message}=Array{Message}(undef, 20))

messages[1] = ruleSPGaussianMeanVarianceOutVPP(nothing, Message(Univariate, PointMass, m=data[:m_s_t_min]),
  Message(Univariate, PointMass, m=data[:v_s_t_min]))
messages[2] = ruleSPGaussianMeanVarianceOutVPP(nothing, Message(Univariate, PointMass, m=data[:m_u][1]),
  Message(Univariate, PointMass, m=data[:v_u][1]))
messages[3] = ruleSPAdditionOutVGG(nothing, messages[1], messages[2])
...

marginals[:u_1] = messages[2].dist * messages[19].dist
marginals[:u_2] = messages[9].dist * messages[20].dist

return marginals

end
```

FIGURE 12 | Snippet of the automatically generated inference algorithm code ($T = 2$) for the Bayesian thermostat. Upon inference, the `step!` function builds an array of messages using pre-derived update rules (e.g., `ruleSPAdditionOutVGG`) and returns posterior beliefs over internal and control states.

5.2. Mountain Car

In this section we simulate an agent that aims to relocate and park itself on a steep hill. However, the agent's engine is too weak to climb the hill directly. Therefore, a successful agent should first climb a neighboring hill, and subsequently use its momentum to overcome the steep incline on the goal-side. This task is also known as the mountain car problem (Friston et al., 2009), which is considered a classical benchmark in the reinforcement learning literature (Sutton and Barto, 2018).

5.2.1. Environmental Process Specification

We start by defining the environmental process, which is similar to the process defined in Ueltzhöffer (2018). We interpret the environmental state $z_t = (\phi_t, \dot{\phi}_t)$, as the respective position and velocity of the mountain car. The horizontal gravitational force component of the hilly landscape depends upon the mountain car's position by [see **Figure 15** (top)]

$$F_g(\phi) = \begin{cases} -0.05\,(2\phi + 1) & \text{if } \phi < 0 \\ -0.05\,\left[(1+5\phi^2)^{-1/2} + \phi^2\,(1+5\phi^2)^{-3/2} + \frac{1}{16}\phi^4\right] & \text{otherwise}\,, \end{cases} \tag{24}$$

Furthermore, we define a velocity-dependent drag as

$$F_f(\dot{\phi}) = -0.1\,\dot{\phi}\,. \tag{25}$$

Through actions, the agent is allowed to set the engine force, which is limited to the interval $(-0.04, 0.04)$, by [see **Figure 15** (bottom)]

$$F_a(a) = 0.04\,\tanh(a)\,. \tag{26}$$

```
(execute, observe)  = initializeWorld() # Let there be a world
(infer, act, slide) = initializeAgent() # Let there be an agent

# Step through the experimental protocol
a = Vector{Float64}(undef, N) # Actions
y = Vector{Float64}(undef, N) # Environmental outcomes
for t = 1:N
    a[t] = act() # Evoke an action from the agent
           execute(a[t]) # The action influences hidden environmental states
    y[t] = observe() # Observe the current environmental outcome (update p)
           infer(a[t], y[t]) # Infer beliefs from current model state (update q)
           slide() # Prepare model for next iteration
end
```

FIGURE 13 | Code for executing the experimental protocol. The `initializeWorld` and `initializeAgent` functions specify closures that return functions for interacting with the environment and agent. This construct allows for encapsulating hidden states in respective scopes for the agent and environment, and allowing only indirect access to environmental states through returned functions.

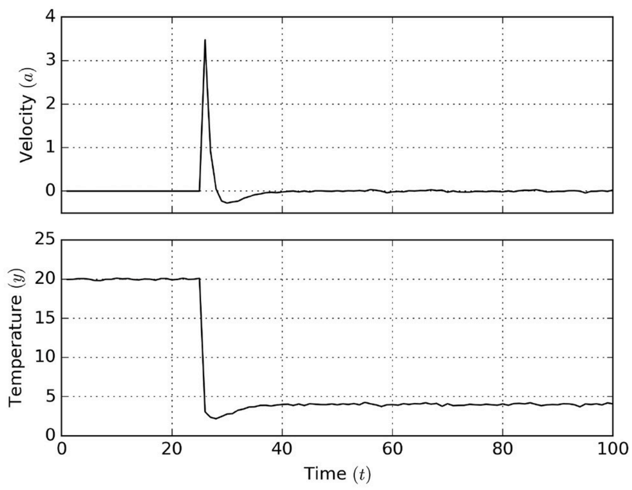

FIGURE 14 | Active inference results for the Bayesian thermostat. After $t = 25$ the agent is allowed to act, moving toward the goal temperature $x_{\ast} = 4$.

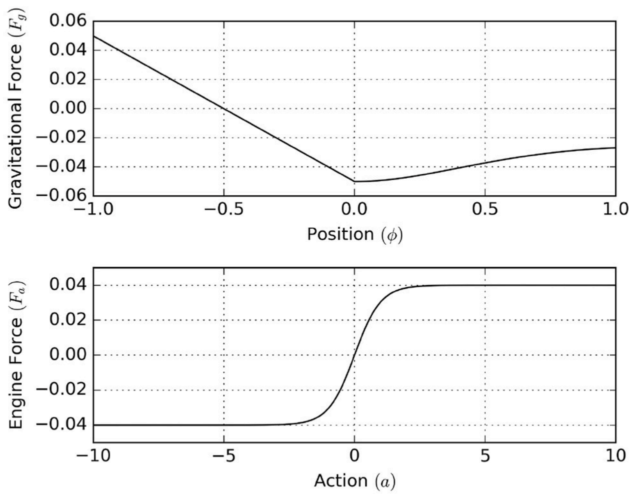

FIGURE 15 | (Top) The horizontal gravitational force component that acts upon the agent as a function of position. **(Bottom)** The horizontal force component of the engine as a function of action.

Assuming unit mass for the mountain car, this leads to the following discrete system dynamics:

$$\dot{\phi}_t = \dot{\phi}_{t-1} + F_g(\phi_{t-1}) + F_f(\dot{\phi}_{t-1}) + F_a(a_t) \qquad (27a)$$
$$\phi_t = \phi_{t-1} + \dot{\phi}_t\,. \qquad (27b)$$

In our simulation, we choose initial position $\phi_0 = -0.5$ and initial velocity $\dot{\phi}_0 = 0$. The environmental process generates outcomes as noisy observations of the current state with an observation noise variance $\theta = 10^{-4} \cdot I$, where I represents the 2×2 identity matrix. This leads to the following environmental system dynamics:

$$z_t = g(z_{t-1}, a_t) \qquad (28a)$$
$$y_t \sim \mathcal{N}(z_t, \theta)\,, \qquad (28b)$$

where Equations (27a, 27b) are summarized by a transition function $g(\cdot)$.

5.2.2. Internal Model Specification

For the internal model we define observation variables $x_k = (\xi_k, \dot{\xi}_k)$, and encode the agent's target to reach a desired state $x_{\ast} = (\xi_{\ast}, \dot{\xi}_{\ast}) = (0.5, 0)$ at time $t = 20$ by defining a time-dependent target prior

$$p'(x_t) = \begin{cases} \mathcal{N}\big(x_t \,|\, 0, 10^{12} \cdot I\big) & \text{if } t < 20 \\ \mathcal{N}\big(x_t \,|\, x_{\ast}, 10^{-4} \cdot I\big) & \text{otherwise}\,. \end{cases} \qquad (29)$$

In other words, we set a vague prior belief over short-term observations ($t < 20$), but aim to reach and remain at the goal state afterwards.

We endow the agent with an accurate model of the environmental dynamics. These dynamics are captured by the transition model for the internal states $s_k = (\zeta_k, \dot{\zeta}_k)$. Because the current state is a non-linear function of the previous state and action, we resort to local linear approximations of the system dynamics as explained in section 3.2. We choose the transition model

$$p(s_k \mid s_{k-1}, u_k) = \mathcal{N}\big(s_k \mid \tilde{g}(s_{k-1}, u_k), 10^{-4} \cdot I\big) \ , \tag{30}$$

where $\tilde{g}$ represents the local linear approximation of g in Equation (28a).

The observation model simply captures the observation noise

$$p(x_k \mid s_k) = \mathcal{N}\big(x_k \mid s_k, 10^{-4} \cdot I\big) \ . \tag{31}$$

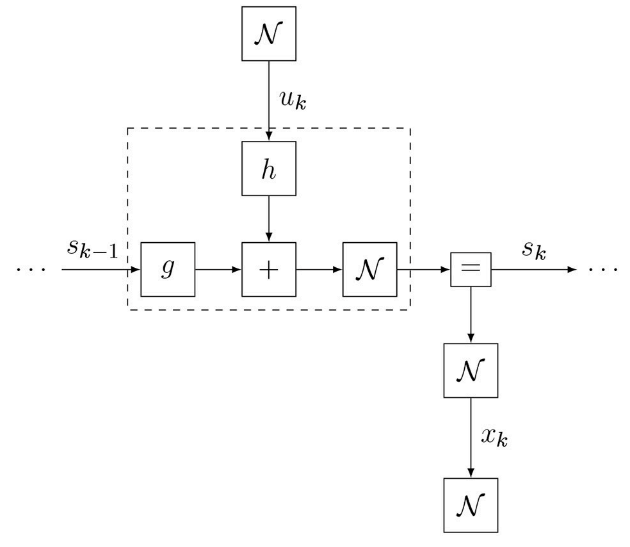

FIGURE 16 | A factor graph visualization of one time slice of the agent's internal model for the mountain car problem (excluding the initial state prior).

The internal model is completed with a tight state prior and vague control priors:

$$p(s_0) = \mathcal{N}\big(s_0 \mid (-0.5, 0), 10^{-12} \cdot I\big) \tag{32a}$$
$$p(u_k) = \mathcal{N}\big(u_k \mid 0, 10^{12}\big) \ . \tag{32b}$$

These priors specify an accurate belief over the initial state and few prior constraints on control signals. **Figure 16** shows one time slice of the agent's internal model.

5.2.3. Simulation Results

The simulation results[1] for 30 time steps and with horizon $T = 20$ are shown in **Figure 17**. Here, a naive policy (right column) with full throttle to the right does not overcome the steepest part of the incline, while the active inference process (left column) infers the need to move left before engaging full throttle to the right in order to reach the desired goal state.

6. RELATED WORK

Languages and toolboxes for automated probabilistic inference are increasingly studied in the research literature under the label Probabilistic Programming (PP). Recent state-of-the-art PP toolboxes such as Stan (Carpenter et al., 2017), Edward (Tran et al., 2016), and Infer.NET (Minka et al., 2018) support a broad spectrum of models and algorithms. However, dynamic models incorporate specific structures that may be exploited for improved algorithm efficiency. The SPM toolbox (Friston, 2014) includes specialized routines for simulating active inference processes, but offers limited modeling flexibility. ForneyLab marries flexible model design with automated derivation of

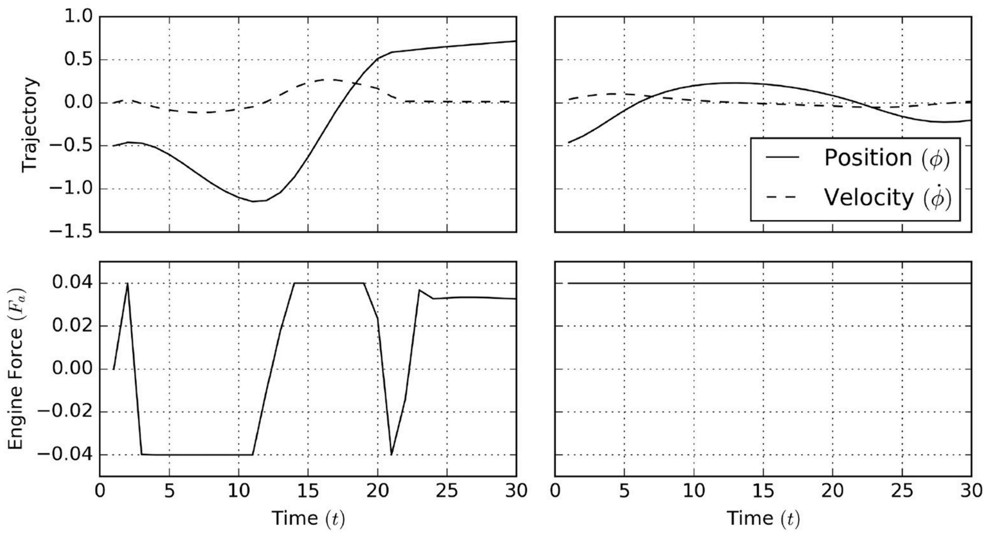

FIGURE 17 | A naive policy in the mountain car task applies a full thrust to the right (right column). This naive policy is unable to make the agent traverse the barrier raised by the gravitational force. In contrast, active inference (left column) automatically infers that the agent should first move left (away from its goal) and subsequently use its momentum (and full forward thrust) to traverse the barrier.

efficient structured inference algorithms on dynamic models (Cox et al., 2019). Furthermore, because ForneyLab produces inference algorithms as stand-alone (Julia) programs, these algorithms can be manually optimized before execution.

Agent-environment interactions can be viewed from the perspective of coupled dynamical systems (Beer, 1995). It was recognized early on that a successful regulating agent must include a model of its environment (Conant and Ashby, 1970). Later efforts in the field of model-predictive control used this idea to build controllers for industrial applications (Camacho and Alba, 2013). Various reinforcement learning techniques have also been applied to the mountain car parking problem (Kuss and Rasmussen, 2004; Fürnkranz et al., 2012; Sutton and Barto, 2018), which can be extended to hierarchical contexts as well (Barto and Mahadevan, 2003). A comparison between active inference, risk-sensitive control (van den Broek et al., 2010) and expected utility maximization is provided by Friston et al. (2015). Further comparisons between active inference and reinforcement learning can be found in Friston (2012), Friston et al. (2012), and Friston and Ao (2012).

While the present paper considers a discrete-time formulation of active inference, simulations of active inference (Friston et al., 2009; Pio-Lopez et al., 2016; Buckley et al., 2017) have also been formulated and performed in the continuous-time domain (Friston et al., 2008). In the continuous-time treatment, preferred states are encoded as attracting sets in the dynamical system, and active inference leads the system to these attracting sets over time. While the continuous-time formulation allows for a more standard mathematical treatment, the discrete-time formulation supports explicit reasoning about targets at specific time-points and thereby more easily supports specification of priors for value-seeking behavior (Friston et al., 2015; Parr and Friston, 2017).

7. DISCUSSION AND CONCLUSION

This paper has described a message passing approach to automating simulations of online active inference processes, together with an experimental protocol that governs the interactions between the agent and its environment. We have tested our protocol on two synthetic applications, namely the Bayesian thermostat and the mountain car parking tasks. Through these examples we have addressed the questions formulated in section 1, and illustrated how:

1. the proposed experimental protocol defines how to simulate the interactions between an active inference agent and its environment (section 4);
2. The ForneyLab toolbox allows for automatic scheduling of message passing algorithms for variational free energy minimization (section 5) in active inference agents.

The FFG formalism offers a modular decomposition of the internal model definition, allowing flexible model adaptations and intuitive visualization of complex models. Moreover, message passing algorithms for free energy minimization can be automatically derived on the FFG formulation of the agent's internal model. Automated derivation with ForneyLab returns the inference algorithm as a Julia program, which can be customized and executed in context of an experimental protocol.

The proposed experimental protocol formulates the active inference process at each time step as an interplay between updating an internal (generative) model with actions and outcomes ("act-execute-observe"), followed by updating the recognition model ("infer") with the (statistical) consequences of the change in the generative model. Crucially, the agent and its environment solely interact through the exchange of actions and outcomes.

A current limitation of active inference with ForneyLab is that high-dimensional models may lead to numerical instabilities. Message passing with improved numerical stability is described by Loeliger et al. (2016). Furthermore, the specific message update order as prescribed by the schedule may have an effect on algorithm convergence. However, little theory still exists on optimal scheduling strategies. An interesting idea was mentioned in de Vries and Friston (2017), where it was suggested to approach the scheduling problem as an inference process that is itself subject to the free energy principle.

The presented approach to active inference relies fully on automatable inference methods. This aproach scales in principle to more complex applications that may be of interest to industry as well. For example, the state space models in the examples can be readily extended to hierarchical generative models (Kiebel et al., 2009; Senoz and de Vries, 2018), which have been shown to be quite powerful in modeling real-world dynamics (e.g., Turner and Sahani, 2008; Mathys et al., 2014).

In order to construct hierarchical models, the policy priors may optionally depend on higher-order states, e.g., $p(u_k|s_k^{(1)})$, which renders prior constraints on control context-dependent. Similarly, target priors may also be made context-dependent, e.g., $p(x_k|s_k^{(1)})$. Contextual processes may thus influence the agent's behavior by modifying prior statistics, which allows the model design engineer to propose hierarchical and context-aware models. For example, when higher-order states evolve over longer timespans, hierarchical nesting leads to deep temporal models (de Vries and Friston, 2017;Friston K. et al., 2017).

Higher-order dynamics could also be learned by free energy minimization (Ramstead et al., 2018). For example, the current simulations internalize a fixed model of the environmental dynamics. By including a prior belief over the dynamics in the internal model, the agent might learn the environmental dynamics from data (Ueltzhöffer, 2018). This adaptive agent then exhibits epistemic behavior and will take action in order to decrease uncertainty about the environmental dynamics (Friston et al., 2016; Cullen et al., 2018). Moreover, the FFG paradigm supports messages that are computed by local gradient or sampling-based methods (Dauwels, 2007), and even allows for learning complex updates from data with the use of amortization techniques (Stuhlmüller et al., 2013; Gershman and Goodman, 2014). With these techniques, an adaptive agent might learn a rich and accurate model of its environment, leading to more effective behavior.

In summary, the current paper has proposed a scalable approach to automatic derivation of active inference algorithms and a practical view on the implementation of simulated active inference systems. We believe that synthetic active inference holds great promise for future engineering applications.

AUTHOR CONTRIBUTIONS

TvdL performed the simulations. TvdL and BdV wrote the manuscript.

FUNDING

This research was made possible by funding from GN Hearing A/S. This work was partly sponsored by research program HearScan with project number 13925, which is (partly) financed by the Netherlands Organization for Scientific Research (NWO).

ACKNOWLEDGMENTS

The authors gratefully acknowledge stimulating discussions with Marco Cox, Ivan Bocharov, Ismail Senoz, and Tjalling Tjalkens of the signal processing systems group at TU Eindhoven and Joris Kraak of GN Hearing. The authors also thank the two reviewers for their valuable comments.

REFERENCES

Attias, H. (1999). "A variational Bayesian framework for graphical models," in *NIPS*, Vol. 12 (Denver, CO) .

Barto, A. G., and Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete Event Dyn. Syst.* 13, 41–77. doi: 10.1023/A:1022140919877

Beer, R. D. (1995). A dynamical systems perspective on agent-environment interaction. *Artif. Intell.* 72, 173–215. doi: 10.1016/0004-3702(94)00005-L

Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. (2017). Julia: a fresh approach to numerical computing. *SIAM Rev.* 59, 65–98. doi: 10.1137/141000671

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. New York, NY: Springer-Verlag Inc.

Buckley, C. L., Kim, C. S., McGregor, S., and Seth, A. K. (2017). The free energy principle for action and perception: a mathematical review. *J. Math. Psychol.* 81, 55–79. doi: 10.1016/j.jmp.2017.09.004

Camacho, E. F., and Alba, C. B. (2013). *Model Predictive Control*. Springer Science & Business Media.

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., et al. (2017). Stan: a probabilistic programming language. *J. Stat. Softw.* 76:1. doi: 10.18637/jss.v076.i01

Conant, R. C., and Ashby, W. R. (1970). Every good regulator of a system must be a model of that system. *Intl. J. Syst. Sci.* 1, 89–97. doi: 10.1080/00207727008920220

Cox, M. (2018). "Robust Expectation propagation in factor graphs involving both continuous and binary variables," in *26th European Signal Processing Conference (EUSIPCO)* (Rome).

Cox, M., van de Laar, T., and de Vries, B. (2018). "ForneyLab.jl: fast and flexible automated inference through message passing in Julia," in *International Conference on Probabilistic Programming* (Boston, MA).

Cox, M., van de Laar, T., and 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

Cullen, M., Davey, B., Friston, K. J., and Moran, R. J. (2018). Active inference in OpenAI gym: a paradigm for computational investigations into psychiatric illness. *Biol. Psychiatry Cogn. Neurosci. Neuroimaging* 3, 809–818. doi: 10.1016/j.bpsc.2018.06.010

Dauwels, J. (2007). "On variational message passing on factor graphs," in *IEEE International Symposium on Information Theory* (Nice), 2546–2550.

Dauwels, J., Korl, S., and Loeliger, H.-A. (2005). "Expectation maximization as message passing," in *International Symposium on Information Theory, ISIT 2005 Proceedings* (Adelaide, SA), 583–586.

de Vries, B., and Friston, K. J. (2017). A factor graph description of deep temporal active inference. *Front. Comput. Neurosci.* 11:95. doi: 10.3389/fncom.2017.00095

Forney, G. (2001). Codes on graphs: normal realizations. *IEEE Trans. Inform. Theory* 47, 520–548. doi: 10.1109/18.910573

Friston K. J. (2014). *SPM12 Toolbox.* Available online at: http://www.fil.ion.ucl.ac.uk/spm/software/

Friston, K. (2009). The free-energy principle: a rough guide to the brain? *Trends Cogn. Sci.* 13, 293–301. doi: 10.1016/j.tics.2009.04.005

Friston, K. (2010). The free-energy principle: a unified brain theory? *Nat. Rev. Neurosci.* 11, 127–138. doi: 10.1038/nrn2787

Friston, K. (2012). "Policies and priors," in *Computational Neuroscience of Drug Addiction*, Springer Series in Computational Neuroscience, eds B. Gutkin and S.H. Ahmed (New York, NY: Springer), 237–283.

Friston, K. (2013). Life as we know it. *J. R. Soc. Interface* 10:20130475. doi: 10.1098/rsif.2013.0475

Friston, K., and Ao, P. (2012). Free energy, value, and attractors. *Comput. Math. Methods Med.* 2012:937860. doi: 10.1155/2012/937860

Friston, K., FitzGerald, T., Rigoli, F., Schwartenbeck, P., O'Doherty, J., and Pezzulo, G. (2016). Active inference and learning. *Neurosci. Biobehav. Rev*. 68, 862–879. doi: 10.1016/j.neubiorev.2016.06.022

Friston, K., and Kiebel, S. (2009). Predictive coding under the free-energy principle. *Philos. Trans. R. Soc. Lond. B Biol. Sci.* 364, 1211–1221. doi: 10.1098/rstb.2008.0300

Friston, K., Kilner, J., and Harrison, L. (2006). A free energy principle for the brain. *J. Physiol.* 100, 70–87. doi: 10.1016/j.jphysparis.2006.10.001

Friston, K., Rigoli, F., Ognibene, D., Mathys, C., Fitzgerald, T., and Pezzulo, G. (2015). Active inference and epistemic value. *Cogn. Neurosci.* 6, 187–214. doi: 10.1080/17588928.2015.1020053

Friston, K., Rosch, R., Parr, T., Price, C., and Bowman, H. (2017). Deep temporal models and active inference. *Neurosci. Biobehav. Rev*. 77, 388–402. doi: 10.1016/j.neubiorev.2017.04.009

Friston, K., Samothrakis, S., and Montague, R. (2012). Active inference and agency: optimal control without cost functions. *Biol. Cybern.* 106, 523–541. doi: 10.1007/s00422-012-0512-8

Friston, K. J., Daunizeau, J., and Kiebel, S. J. (2009). Reinforcement learning or active inference? *PLoS ONE* 4:e6421. doi: 10.1371/journal.pone.0006421

Friston, K. J., Parr, T., and de Vries, B. (2017). The graphical brain: belief propagation and active inference. *Netw. Neurosci.* 1, 381–414. doi: 10.1162/NETN_a_00018

Friston, K. J., Trujillo-Barreto, N., and Daunizeau, J. (2008). DEM: a variational treatment of dynamic systems. *Neuroimage* 41, 849–885. doi: 10.1016/j.neuroimage.2008.02.054

Fürnkranz, J., Hüllermeier, E., Cheng, W., and Park, S.-H. (2012). Preference-based reinforcement learning: a formal framework and a policy iteration algorithm. *Mach. Learn.* 89, 123–156. doi: 10.1007/s10994-012-5313-8

Gershman, S., and Goodman, N. (2014). "Amortized inference in probabilistic reasoning," *Proceedings of the Cognitive Science Society*.

Huang, Y., and Rao, R. P. N. (2011). Predictive coding. *Wiley Interdisc. Rev. Cogn. Sci.* 2, 580–593. doi: 10.1002/wcs.142

Kiebel, S. J., Daunizeau, J., and Friston, K. J. (2009). Perception and hierarchical dynamics. *Front. Neuroinform.* 3:20. doi: 10.3389/neuro.11.020.2009

Koller, D., and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Korl, S. (2005). *A Factor Graph Approach to Signal Modelling, System Identification and Filtering*. Ph.D. thesis, Swiss Federal Institute of Technology, Zurich.

Kuss, M., and Rasmussen, C. E. (2004). "Gaussian processes in reinforcement learning," in *Advances in Neural Information Processing Systems* (Vancouver, BC), 751–758.

Loeliger, H.-A. (2004). An introduction to factor graphs. *IEEE Signal Process. Mag.* 21, 28–41. doi: 10.1109/MSP.2004.1267047

Loeliger, H.-A., Bruderer, L., Malmberg, H., Wadehn, F., and Zalmai, N. (2016). On sparsity by NUV-EM, Gaussian message passing, and Kalman smoothing. *[arXiv preprint]. arXiv:1602.02673 [cs, math]*. doi: 10.1109/ITA.2016.7888168

Loeliger, H.-A., Dauwels, J., Hu, J., Korl, S., Ping, L., and Kschischang, F. R. (2007). The factor graph approach to model-based signal processing. *Proc. IEEE* 95, 1295–1322. doi: 10.1109/JPROC.2007.896497

Mathys, C. D., Lomakina, E. I., Daunizeau, J., Iglesias, S., Brodersen, K. H., Friston, K. J., et al. (2014). Uncertainty in perception and the Hierarchical Gaussian Filter. *Front. Hum. Neurosci.* 8:825. doi: 10.3389/fnhum.2014.00825

Minka, T., Winn, J., Guiver, J., Zaykov, Y., Fabian, D., and Bronskill, J. (2018). *Infer.NET 2.7.* Microsoft Research Cambridge.

Parr, T., and Friston, K. J. (2017). Uncertainty, epistemics and active inference. *J. R. Soc. Interface* 14:20170376. doi: 10.1098/rsif.2017.0376

Parr, T., and Friston, K. J. (2018). Generalised free energy and active inference: can the future cause the past? *BioRxiv [preprint]*. doi: 10.1101/304782

Pio-Lopez, L., Nizard, A., Friston, K., and Pezzulo, G. (2016). Active inference and robot control: a case study. *J. R. Soc. Interface* 13:20160616. doi: 10.1098/rsif.2016.0616

Ramstead, M. J. D., Badcock, P. B., and Friston, K. J. (2018). Answering Schrödinger's question: a free-energy formulation. *Phys. Life Rev*. 24, 1–16. doi: 10.1016/j.plrev.2017.09.001

Rao, R. P. N., and Ballard, D. H. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nat. Neurosci.* 2, 79–87. doi: 10.1038/4580

Reller, C. (2012). *State-Space Methods in Statistical Signal Processing: New Ideas and Applications*. Ph.D. thesis, ETH Zurich.

Senoz, I., and de Vries, B. (2018). "Online variational message passing in the hierarchical Gaussian filter," in *2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP)* (Aalborg), 1–6.

Stuhlmüller, A., Taylor, J., and Goodman, N. (2013). "Learning stochastic inverses," in *Advances in Neural Information Processing Systems* (Lake Tahoe, NV), 3048–3056.

Sutton, R. S., and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT Press.

Tran, D., Kucukelbir, A., Dieng, A. B., Rudolph, M., Liang, D., and Blei, D. M. (2016). Edward: a library for probabilistic modeling, inference, and criticism. *[arXiv preprint]. arXiv:1610.09787*.

Turner, R., and Sahani, M. (2008). "Modeling natural sounds with modulation cascade processes," in *Advances in Neural Information Processing Systems (NIPS)* (Telluride, CO).

Ueltzhöffer, K. (2018). Deep active inference. *Biol. Cybern. [arXiv preprint]. arXiv:1709.02341*. doi: 10.1007/s00422-018-0785-7

van de Laar, T., Cox, M., Senoz, I., Bocharov, I., and de Vries, B. (2018). "ForneyLab: a toolbox for biologically plausible free energy minimization in dynamic neural models," in *Conference on Complex Systems (CCS)* (Thessaloniki).

van den Broek, B., Wiegerinck, W., and Kappen, B. (2010). "Risk sensitive path integral control," in *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence UAI'10* (Arlington, VA: AUAI Press), 615–622.

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.