Daniel Brito dos Santos

# PingPOMDP: a Computational System to Experiment with Active Inference

Campos dos Goytacazes

2023

# 1 Introduction

Active Inference (AIF) is a theoretical framework to explain how agents perceive and act in the world. Karl Frinston defines it as a "way of understanding sentient behavior" (Parr; Pezzulo; Friston, 2022). "Sentient" here means any system capable of adaptively interacting with its environment (Kagan et al., 2022).

AIF suggests that any sentient system learns a probabilistic model of their habitable environment and that the states of the system change to maximize the evidence for this model [11], [12]. The resulting scheme casts perception, action, and learning as emergent processes of approximate Bayesian inference, thereby offering a potentially unifying theory of adaptive biological or artificial systems (Tschantz et al., 2020).

The applications of this theory span from controlling robotic arms (Baltieri; Buckley, 2019) to modeling schizophrenia and its response to antipsychotics (Adams et al., 2022).

Recently, a remarkable study by Kagan et al. (2022) demonstrated that *in vitro* neurons can learn and exhibit sentience when embodied in a simulated game world. They used a high-density electrode grid to interface the neurons with a version of the arcade game "Pong", where the neurons could control the paddle and receive feedback from the game. Applying implications from the theory of AIF, the electrodes administered unpredictable feedback to undesired behavior (losing the ball), and vice-versa (predictable feedback when hitting the ball). They called this system "DishBrain". The results showed apparent learning within five minutes of real-time gameplay. The neurons self-organized their activity and structure to maximize the score and avoid losing. The researchers termed "synthetic biological intelligence" this goal-directed response to sparse sensory information about the consequences of their actions.

This study inspired me to pursue a similar project, but using a computational model of AIF instead of *in vitro* neurons. My goal is to develop a computational structure to interface an AIF model with "Pong", inspired by the electrode grid and feedback protocol used in the "DishBrain system". I will compare the performance and behavior of my implementation with Kagan et al. (2022) results. The aim is to establish possible metrics and computational infrastructure to compare different models and environments through this interface that can possibly map to experiments with *in vitro* neuron cultures.

## 1.1 Problem statement

The study by Kagan et al. (2022) showed that *in vitro* neurons can learn and exhibit sentience when embodied in a simulated game world using a feedback protocol inspired by

the theory of Active Inference (AIF). This significant finding raises a compelling question: can a computational model of AIF achieve similar outcomes when connected to the same game environment?

The existing challenge lies in the absence of a computational framework that can effectively connect an AIF model with a modified Pong game, emulating the electrode grid employed in the *DishBrain* system. Furthermore, there is a lack of established metrics and computational infrastructure to facilitate the comparison of various models and environments through this interface. Thus, the objective of this project is to address these gaps by developing and assessing a suitable computational structure using Python and the pymdp library. By doing so, we can further investigate the potential of AIF in achieving similar outcomes within a game environment.

## 1.2 Hypothesis

The central research hypothesis of this work is that a computational system inspired by the *DishBrain* system interface with an AIF model instead of a culture of neurons will exhibit comparable learning and performance in the modified Pong game. This hypothesis will be tested by measuring the score, reaction time, and accuracy of the system and comparing them with those of Kagan et al. (2022).

## 1.3 Objectives

The general objective of this project is to demonstrate that a computational system inspired by the *DishBrain* system interface with an AIF model instead of a culture of neurons can exhibit "synthetic biological intelligence" in the modified Pong game.

The specific objectives are:

- To develop a computational system that integrates an AIF model and a modified Pong game environment using the pymdp library and Python.
- To evaluate the learning and performance of the system in the game using score, reaction time, and accuracy as metrics.
- To compare the results with those of Kagan et al. (2022) using statistical tests and visualizations and discuss the implications for synthetic biological intelligence research.

## 1.4 Scope and limitations

The scope of this project is restricted to implementing a simple AIF model that can learn to control the paddle in Pong through a grid electrode interface, as close as possible to the specifications of Kagan et al. (2022)'s *DishBrain* system. The project will not attempt to replicate all the details and nuances of their study, nor to explore more complex or realistic scenarios.

## 1.5 Methodology

This project will adopt an experimental research design to test the hypothesis that a computational system inspired by the *DishBrain* system interface with an AIF model instead of a culture of neurons will exhibit comparable learning and performance in the modified Pong game. The methodology will consist of the following steps:

1. Implementing an AIF model using the pymdp library (Heins et al., 2022a), which is a Python framework for building and simulating AIF models. The model will consist of a generative model that represents the beliefs about the game states and actions, and an inference algorithm that updates these beliefs based on sensory inputs and prediction errors.

2. Implementing the modified Pong game environment using Python, similar to the one used by Kagan et al. (2022).

3. Implementing a computational interface inspired by the electrode grid in the DishBrain system that connects the model and the environment. The interface will use a high-density array of electrodes to stimulate and record the activity of the model. The electrodes will administer predictable and unpredictable stimuli depending on whether the model hits the ball or not. The stimuli will be encoded as sensory inputs that affect the prediction errors of the model.

4. Running experiments with different model parameters and game settings. The experiments will vary the parameters of the model (such as learning rate, precision, prior beliefs) and the settings of the game (such as ball speed, paddle size, feedback frequency) to observe how they affect the learning and performance of the system. Each experiment will consist of 10 trials of 5 minutes each.

5. Measuring the score, reaction time, and accuracy of the system in the game. The score will be calculated as the number of hits minus the number of misses. The reaction time will be calculated as the time elapsed between the ball crossing a threshold distance from the paddle and the paddle moving to hit it. The accuracy will be calculated as the proportion of hits over total attempts.

6. Comparing the results with those of Kagan et al. (2022) using statistical tests and visualizations. The results will be analyzed using descriptive statistics (such as mean, standard deviation, range) and inferential statistics (such as t-tests, ANOVA, correlation) to test for significant differences and relationships between the system and the culture of neurons. The results will also be visualized using graphs (such as line charts, bar charts, scatter plots) to illustrate the trends and patterns of learning and performance.

7. Discussing the implications for AIF research. The discussion will interpret the findings in relation to the hypothesis and research question. It will also compare and contrast the computational system with the biological system in terms of their behavior in the game environment. It will highlight the strengths and limitations of the system and suggest possible directions for future research.

The limitations of this project are mainly related to the validity and generalizability of the results. The validity may be affected by factors such as measurement error, confounding variables, and external noise. The generalizability may be limited by the small sample size, the artificial nature of the game environment, and the differences between the computational model and the biological system. These limitations will be addressed by using appropriate methods of data collection and analysis, controlling for extraneous factors, and acknowledging the scope and assumptions of the research.

## 1.6 Structure

The structure of this monograph is as follows:

- Chapter 1: This chapter introduces the research topic, problem, hypothesis, objectives, scope and limitations, methodology, and structure of the monograph.

- Chapter 2: Theoretical background. This chapter introduces the concepts and principles of active inference and its computational implementation. It also reviews the recent literature on active inference, especially the paper by Kagan et al. (2022). It provides the theoretical and conceptual background for the research and identifies the gap in the existing knowledge that the research aims to fill.

- Chapter 3: Modeling. This chapter describes the implementation of the PingPOMDP system and its components: the active inference model, the Pong game environment, and the GridLink interface.

- Chapter 4: Experimentation and analysis. This chapter reports the results of the experiments and analyzes them using descriptive statistics and visualizations.

- Chapter 5: Conclusion. This chapter discusses the main findings, implications, limitations, and future work of this project.

# 2 Active Inference and its Computational Implementation

How do living organisms persist while engaging in adaptive exchanges with their environment? This is the main question that Active Inference (AIF) seeks to address. Living organisms constantly engage in reciprocal interactions with their environment (including other organisms). They emit *actions* that change the environment and receive *sensory* observations from it as schematically illustrated in Figure 1.

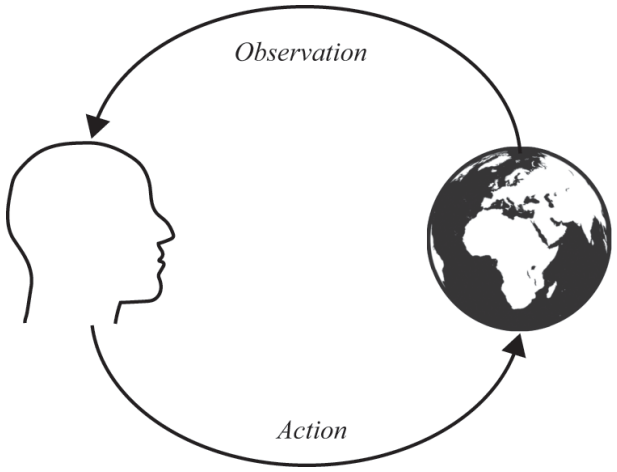


Figure 1 – An action-perception cycle reciprocally connecting a creature and its environment source: (Parr; Pezzulo; Friston, 2022)

These organisms can only maintain their integrity by exerting adaptive control over this action-perception loop. AIF is a normative solution to this problem, It helps to understand *what* organisms must do to keep existing (minimize their free energy) and *why* (to minimize the surprise of their sensory observations). This means acting to solicit sensory observations that correspond to desired outcomes (e.g., the sensations that accompany secure nutrients and shelter). (Parr; Pezzulo; Friston, 2022)

AIF proposes that despite their diverse manifestations, the central aspects of behavior, cognition, and adaptation in living organisms are amenable to a coherent explanation from first principles.(Parr; Pezzulo; Friston, 2022)

On the other hand, we can also arrive at AIF from the notion of the Bayesian brain. This theory casts the brain as an inference engine trying to optimize probabilistic representations of the causes of its sensory input. It then motivates AIF as a specific, variational approximation to the otherwise intractable inferential problem. This perspective is useful to illustrate *how* AIF agents minimize their free energy, which presents AIF as

not just a principle but also a mechanistic explanation, or process theory of cognitive functions. (Parr; Pezzulo; Friston, 2022)

Figure 2 illustrates these two complementary perspectives on AIF as well as their theoretical foundations on what Parr, Pezzulo and Friston (2022) calls "the two roads to Active Inference".

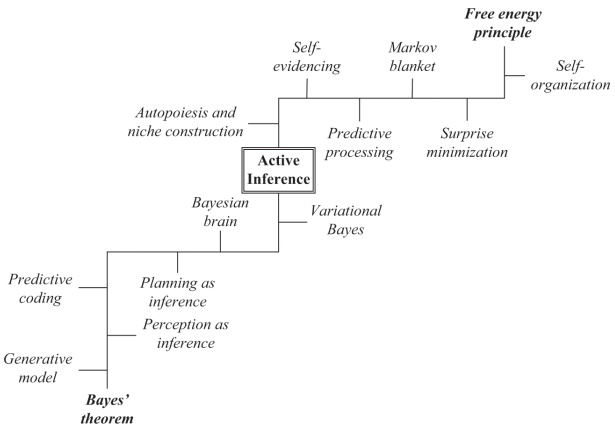


Figure 2 – Two roads to Active Inference. source: (Parr; Pezzulo; Friston, 2022)

Thereby, AIF postulates that all facets of behavior and cognition follow a unique imperative: minimizing the surprise of their sensory observations. Surprise here is the measure of how much an agent's current sensory observations differ from its preferred sensory observations. However, minimizing surprise is not something that can be done by passively observing, but rather by controlling the action-perception loop to solicit desired sensory observations. Minimizing surprise turns out to be a challenging problem. AIF offers a solution to this problem. It assumes that even if living organisms cannot directly minimize their surprise they can minimize a proxy called (variational) *free energy*. This quantity can be minimized through computation in response to and in anticipation of sensory observations. (Parr; Pezzulo; Friston, 2022)

In this regard, AIF's process theory postulates that all these phenomena may be seen as optimizing two complementary objective functions: a variational free energy (VFE), which measures the fit between an internal model and past sensory observations, and an expected free energy (EFE), which scores possible future courses of action about prior preferences (Costa et al., 2020).

Crucially, in AIF, preferences and goals are encoded as priors in the generative model. That is, agents may incorporate an optimism bias scoring "preferred" sensations as more likely. This lends a higher plausibility to those courses of action that realize these sensations (Costa et al., 2020; Parr; Pezzulo; Friston, 2022; Heins et al., 2022a; Smith; Badcock; Friston, 2021).

Active inference can thus be framed as the minimization of surprise by perception and action. This means agents select from different possible courses of action (i.e., policies)

in order to realize their preferences and thus minimize the surprise that they expect to encounter in the future. This enables a Bayesian formulation of the perception–action cycle: (0) agents perceive the world by minimizing variational free energy (VFE), ensuring their model is consistent with past observations, and (2) act by minimizing expected free energy (EFE), to make future sensations consistent with their model. This account of behavior can be concisely framed as **self-evidencing**. (Costa et al., 2020):

## 2.1 Active Inference as POMDP

Our first premise is that agents represent the world through an internal model. This probabilistic model and the probabilistic beliefs that it encodes are continuously updated to mirror the environment and its dynamics. Such a world model is considered to be generative because it is able to generate predictions about sensations, given beliefs about future states of being. (Costa et al., 2020)

Formally, the generative model is a joint probability distribution that specifies how hidden states cause sensory consequences. We'll see that minimization of variational free energy enables to "invert" the model, that is, determine the most likely hidden states given sensations. (Costa et al., 2020)

Following the literature (Tschantz et al., 2020; Smith; Friston; Whyte, 2021; Costa et al., 2020; Heins et al., 2022a), we consider the formalization of AIF in the context of a partially observed Markov decision process (POMDP) discrete in time and space:

> At each time step $t$, the true state of the environment $\hat{s}_t \in \mathbb{R}^{d_{\hat{s}}}$ evolves according to the stochastic transition dynamics $\hat{s}_t \sim p(\hat{s}_t|\hat{s}_{t-1}, a_{t-1})$ where $a \in \mathbb{R}^{d_a}$ denotes an agent's actions. We consider agents that do not have access to the true state of the environment, but instead, receive observations $o_t \in \mathbb{R}^{d_o}$ generated according to $o_t \sim p(o_t|\hat{s}_t)$. As such, agents must operate on *beliefs* $s_t \in \mathbb{R}^{d_s}$ about the true state of the environment $\hat{s}_t$. We denote the true dynamics with upright letters $\mathrm{p}(\cdot)$ and a model of these dynamics (the agent) with italics $p(\cdot)$. (Tschantz et al., 2020)

AIF proposes that agents implement and update a generative model of their world, a joint probability distribution over sensory data and the hidden (or latent) causes of these data. That is, $p(\tilde{o}, \tilde{s}, \pi, \theta)$ over observations $o$, hidden states $s$, control states $\pi$ and parameters $\theta$. where the tilde notation denotes a sequence of variables through time, $\pi$ denotes a *policy* (a sequence of actions), $\pi = \{a_0, \ldots, a_T\}$, and $\theta \in \Theta$ denotes parameters of the generative model, which are themselves random variables. (Heins et al., 2022a; Tschantz et al., 2020)

Additionally, agents maintain a recognition distribution $q(\hat{s}, \pi, \theta)$, representing an agents (approximately optimal) beliefs over states $\hat{s}$, policies $\pi$ and model parameters $\theta \in \Theta$. As new observations are sampled, agents update the parameters of their recognition distribution to minimize variational *free energy* (VFE) $\mathcal{F}$ (Tschantz et al., 2020):

$$\mathcal{F}(\hat{o}) = \mathbb{E}_{q(\hat{s},\pi,\theta)}[\ln q(\hat{s}, \pi, \theta) - \ln p(\tilde{o}, \tilde{s}, \pi, \theta)] \tag{2.1}$$

This makes the recognition distribution $q(\hat{s}, \pi, \theta)$ converge towards an approximation of the (intractable) posterior distribution $p(\tilde{s}, \pi, \theta|\tilde{o})$, thereby implementing a tractable form of approximate Bayesian inference [19]. Free energy then provides a proxy for how surprising (i.e., unlikely) some observations are under the agent's model. (Tschantz et al., 2020)

While minimizing Equation 2.1 provides an estimate for how surprising some observations are, it cannot reduce this quantity directly. To achieve this, agents must change their observations through action. Acting to minimize VFE ensures the minimization of *surprisal* $(-\ln p(\tilde{o}))$, or the maximization of the Bayesian *model evidence* $p(\tilde{o})$ since free energy provides an upper bound on *surprisal*. (Tschantz et al., 2020)

Active inference, therefore, proposes that agents select policies in order to minimize *expected* free energy $\mathcal{G}$ [12] [33] where the expected free energy for a given policy $\pi$ at some time $\tau$ is:

$$\mathcal{G}(\pi, \tau) = \mathbb{E}_{q(o_\tau,s_\tau,\theta|\pi)}[\ln q(s_\tau, \theta|\pi) - \ln p(o_\tau, s_\tau, \theta|\pi)] \tag{2.2}$$

Selecting policies that minimize Equation 2.1 will ensure that probable (i.e. favorable, given an agent's normative priors) observations are preferentially sampled, while also ensuring that agents gather information about their environment. Because (Tschantz et al., 2020):

- AIF proposes that an agent's **goals and desires** are encoded in the generative model as **prior preferences** for favorable observations.
- Expected free energy can be decomposed into *extrinsic* value, which quantifies the degree to which expected observations are congruent with an agent's prior beliefs, and intrinsic value, which quantifies the amount of information an agent expects to gain from enacting some policy [10], [11], [13].

Incorporating perception, planning and decision-making. This formalises the action–perception cycle: (1) an agent is presented with a stimulus, (2) it infers its latent

causes, (3) plans into the future and (4) realises its preferred course of action. (Costa et al., 2020)

## 2.2 Computational Implementation of the Active Inference POMDP

Usually in AIF implementations, the agents hold beliefs about (Heins et al., 2022a; Costa et al., 2020; Tschantz et al., 2020):

- the probability $P(s_1)$ of the initial state (specified as $\mathbf{D}$),
- the transition probabilities $P(s_\tau|s_{\tau-1}, u_{\tau-1})$ from one state to the next (defined as the transition likelihood matrix $\mathbf{B}$),
- and the probability $P(o_\tau|s_\tau)$ of outcomes given states (the observation likelihood matrix $\mathbf{A}$).

Additionally:

- Control states can be formally related to policies by writing down an additional likelihood, a "policy-to-control" mapping $p(a_t|\pi)$, that links a given policy to the control state it entails at time $t$.
- Agents also perform *inference* about *policies*, which naturally entails goal-directed and uncertainty-resolving behavior.
- Additional parameters are captured as a single vector of hyperparameters $\theta$ which might correspond to the parameters of Dirichlet priors over likelihood distributions $p(o_t|s_t)$ and $p(s_t|s_{t-1}, \pi)$, for instance.
- Inference about these hyperparameters is often assumed to occur on a slower timescale than inference about hidden states and policies. Therefore, this process is referred in the AIF literature as learning.

Thereby, to simulate agents equipped with POMDP generative models, the joint distribution $p(\tilde{o}, \tilde{s}, \pi, \theta)$ further factorizes into a set of categorical and Dirichlet distributions: the likelihoods and priors of the generative model. This general expression can be expressed as (Heins et al., 2022b):

$$P(o_{1:T}, s_{1:T}, \pi, \theta) = P(\theta)P(s_1)P(\pi)\prod_{\tau=2}^{T} P(s_\tau|s_{\tau-1}, \pi; \theta)\prod_{\tau=1}^{T} P(o_\tau|s_\tau; \theta) \tag{2.3}$$

Where:

- The observation likelihood $P(o_\tau|s_\tau)$ represents the agent's beliefs about how hidden states $s$ generate observations $o$.
- The transition model $P(s_\tau|s_{\tau-1}, u_{\tau-1})$ represents the agent's beliefs about how hidden states at time $\tau - 1$ cause hidden states at the next time $\tau$, conditioned on some control states (actions) $u_{\tau-1}$.
- The prior over initial hidden states $P(s_1)$ represents the agent's baseline belief.
- The prior distribution over observations $P(o_{1:\tau})$ specifies an agent's goals as a desired distribution over observations.

### 2.2.1 The pymdp model (Heins et al., 2022a)

pymdp represents the distributions of the generative model by using categorical distributions, which assign a probability value between 0 and 1 to each discrete outcome level of the distribution sample space. This representation is natural considering that our focus is on generative models of discrete states that evolve in discrete time. Meaning that there is an integer number of discrete levels of both the states of the environment and of the observations.

Mathematically, we refer to categorical distributions with the notation $p(x) = Cat(\phi), \phi \in \{z \in \mathbb{R}^n | z_i > 0, \sum_i z_i = 1\}$. Meaning that the distribution over the random variable $x$ is described by a categorical distribution with an n-dimensional vector of parameters $\phi$, where $n$ is the cardinality of the sample space of $X$.

pymdp represents these distributions numerically as multidimensional arrays that contain their parameters. They can be:

- Vector-valued marginal distributions (e.g. $p(x)$) usually playing the role of priors, encoded as simple 1-D vectors.
- Conditional categorical distributions (e.g. $p(y|x)$ in the form of NDarrays (matrices and tensors) playing the role of likelihoods in the generative model. Conditional categorical distributions are just collections of 1-D categorical vectors, with as many 1-D vectors as there are levels of the conditioning variable.

The core data structure for representing arrays is the Python array programming library numpy [46].

For example, pymdp encodes some discrete distribution $P(y|x)$ as a matrix of size $N \times M$ where N is the number of levels of the support variable $y$ and M is the number of levels of the conditioning variable $x$.

Conditional distributions in pymdp can be expressed as: the first dimensions of the matrix or NDarray represent the support of the conditional distribution, while the lagging dimensions represent the random variables being conditioned on.

In pymdp notation, the observation likelihood $P(o_\tau|s_\tau)$ is constructed as the $\mathbf{A}$ array. In simple models, $\mathbf{A}$ will be a $O \times S$ matrix, where $O$ is the number of outcomes or levels of the observations $o$ and $S$ is the number of levels of hidden sates $s$. The $\mathbf{A}[i, j]$ entry encodes the probability of seeing observation $i$, given state $j$. That is, each column $\mathbf{A}[:, j]$ stores a vector of categorical parameters that encodes the distribution $P(o_\tau|s_{\tau=s_j})$

Similarly the transition likelihood $P(s_\tau|s_{\tau-1}, u_{\tau-1})$ is constructed as the $\mathbf{B}$ array, which is a $S \times S \times U$ NDarray, where $U$ is the number of levels og the control states and entry $\mathbf{B}[i, j, k]$ encodes the probability transitioning to state $i$ from state $j$ when control state or action $k$ is taken by the agent at time $t - 1$.

One can also specify an initial prior over states $p(s_1)$, called the D vector (of length S) that represents the agent's beliefs about the distribution over hidden states at the first timestep of the time horizon $\tau = 1$.

Finally, in order to achieve goal-directed behavior under AIF, it is necessary to represent some desired state or goal into the generative model. In AIF we specify a prior distribution over observations, also known as the "prior preferences". Then, inference over control states is biased by these preference distributions, leading agents to choose actions that bring them to states expected to lead to preferred observations. In pymdp this is represented by the $\mathbf{C}$ array of length $O$. This vector can optionally be time-dependent to represent goals that change over time, or desire to reach a specific goal in a time-dependent manner.

After specifying the generative model in terms of its likelihood and prior distributions one can build an active inference agent using the Agent() constructor. $\mathbf{A}$ and $\mathbf{B}$ arrays are mandatory, while $\mathbf{C}$ and $\mathbf{D}$ vectors are optional inputs to the constructor (the defaults are uniform distribution).

## 2.3 Related works

In this section, we review relevant works that are closely related to the topic of Active Inference (AIF) and its applications. The studies discussed here provide a foundation for understanding the theoretical framework of AIF, its practical implementations, and its potential impact on various domains.

### 2.3.1 PingPOMDP foundations

- Kagan et al. (2022). **In vitro neurons learn and exhibit sentience when embodied in a simulated game-world**. Kagan et al.'s work developed DishBrain, a system to integrate *in vitro* neurons with a simulated game-world mimicking the arcade game "Pong" and applying implications from the theory of AIF. They demonstrated that neurons can exhibit adaptive and goal-directed behavior and self-organize their activity and structure in response to sparse sensory information about the consequences of their actions given by the DishBrain System. This system is a direct inspiration for PingPOMDP.

- Heins et al. (2022a). **pymdp: A Python library for active inference in discrete state spaces.**

  Heins et al.'s work introduces pymdp, a Python library for simulating Active Inference agents with discrete state-space generative models. The paper explains the theoretical, mathematical, and computational principles behind the library, as well as its motivation and advantages. They also demonstrate how the library can be used to design and test Active Inference models for various domains and tasks. The paper also acknowledges how pymdp builds on and improves upon the DEM toolbox, the original and widely-used MATLAB implementation of Active Inference in SPM. While DEM is the gold standard in Active Inference literature, it has some limitations such as being expensive, monolithic, and domain-specific. pymdp overcomes these limitations by being free, modular, and generic. By using Python, the most popular programming language in recent research, pymdp offers more accessibility, flexibility, and compatibility than DEM.

  This library, and its related technical paper, will support the implementation of the PingPOMDP model in this project. By relying on a robust and trusted implementation of Active Inference, I can focus on developing the interface and the game environment and exploring the effects of different parameters and configurations on the learning and behavior of the agent.

### 2.3.2 Active Inference Foundations

- Costa et al. (2020). **Active inference on discrete state-spaces: A synthesis**. Da Costa et al.'s work provides a comprehensive review of the theory and methods of Active Inference on discrete state-spaces. The paper covers the main concepts and equations of Active Inference, as well as its relation to other frameworks such as reinforcement learning, Bayesian inference, and information theory. They also discuss some of the challenges and open questions in Active Inference research.

- Parr, Pezzulo and Friston (2022). **Active Inference: The Free Energy Principle in Mind, Brain, and Behavior**. Parr et al.'s work offers an introduction and overview of Active Inference as a general theory of mind, brain, and behavior. The paper explains the key principles and assumptions of Active Inference, as well as its implications for understanding various aspects of cognition and action. They also provide examples of how Active Inference can be applied to model different phenomena such as perception, learning, decision making, and mental disorders.

- Tschantz et al. (2020). **Scaling Active Inference**. Tschantz et al.'s work explores the scalability and applicability of Active Inference across different levels of abstraction and complexity. The paper proposes a hierarchical formulation of Active Inference that can handle large state-spaces and long time horizons. They also demonstrate how Active Inference can be implemented using deep neural networks and applied to complex tasks such as image classification and reinforcement learning.

- Smith, Friston and Whyte (2021). **A Step-by-Step Tutorial on Active Inference and its Application to Empirical Data**. Smith et al.'s work provides a step-by-step tutorial on how to apply Active Inference to empirical data. The paper guides the reader through the process of building an Active Inference model, fitting it to behavioral data, testing its predictions, and interpreting its results. They also illustrate how Active Inference can be used to address different research questions such as model comparison, parameter estimation, hypothesis testing, and model selection.

### 2.3.3 Applications

- Smith, Badcock and Friston (2021). **Recent advances in the application of predictive coding and active inference models within clinical neuroscience**. Smith et al.'s work reviews the recent advances in applying predictive coding and active inference models to clinical neuroscience. The paper discusses how these models can help explain the neural mechanisms underlying various psychiatric disorders such as schizophrenia, autism, depression, anxiety, and addiction. They also highlight the potential of these models for developing new diagnostic tools and therapeutic interventions.

- Adams et al. (2022). **Everything is connected: Inference and attractors in delusions**. Adams et al.'s work uses active inference to explain the formation and maintenance of delusions. The paper argues that delusions can be seen as abnormal beliefs that arise from faulty inference processes in the brain. They show how delusions can be modeled as attractors in a dynamical system, and how they can be influenced by various factors such as sensory evidence, prior beliefs, and social feedback.

These selected works provide a comprehensive understanding of the theoretical foundations of AIF and its practical applications in various domains. They also serve as valuable references for the current research project, which aims to develop a computational model of AIF interfaced with the game "Pong." The comparison of this implementation with the *in vitro* neuron-based "DishBrain" system described by Kagan will, hopefully, contribute to the establishment of metrics and computational infrastructure for future experiments with *in vitro* neuron cultures.

# 3 Modeling of the "PingPOMDP" system

The main goal of this project is to show that a computational system inspired by the *DishBrain* system (Kagan et al., 2022) interfaced with an AIF model instead of a culture of neurons can display similar "synthetic biological intelligence" in a similar game environment with a similar interface.

This computational system, called "PingPOMDP", consists of an AIF model connected to a modified Pong game environment by a new interface named "GridLink". As Figure 3 illustrates, GridLink converts the Pong environment states into observations for the agent, and the agent's actions into Pong environment actions (namely paddle_up or paddle_down). The idea is that the model begins with random beliefs and, as it interacts with the environment, it updates its internal model and improves its actions, thus exhibiting learning and "sentience" as it adapts its internal structure to the environment. We will investigate how this learning depends on the GridLink and model parameters, and how our computational abstraction relates to the DishBrain system with its neuron cultures.

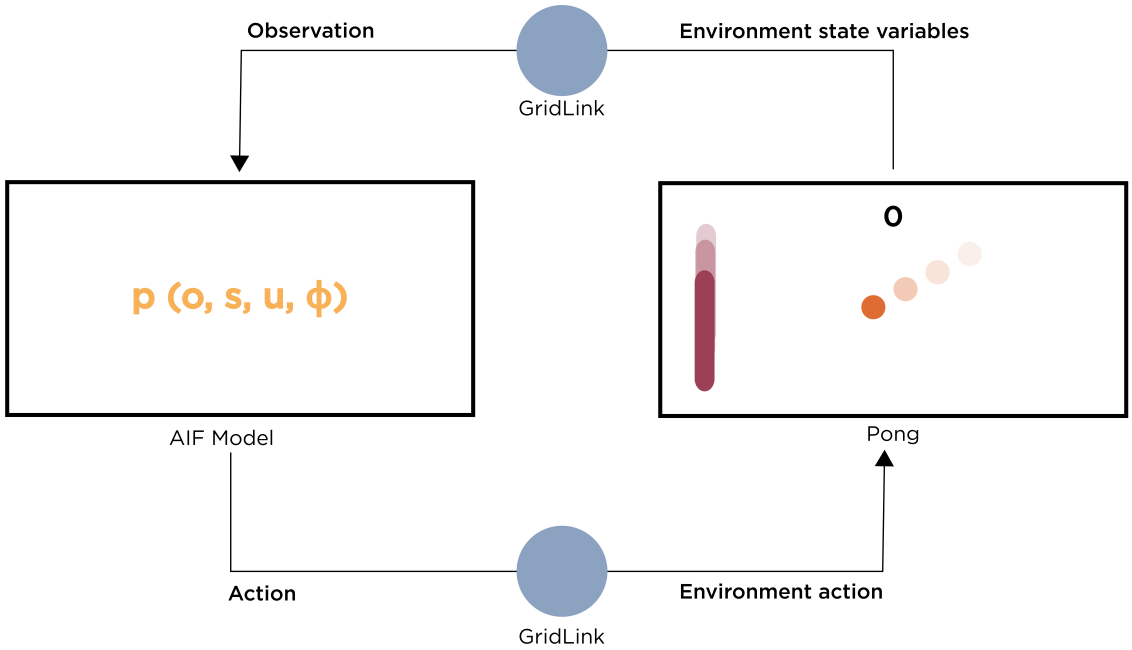


Figure 3 – PingPOMDP conceptual diagram

Moreover, the comparison between DishBrain and PingPOMDP may suggest that our system is a reliable *in silico* simulation of *in vitro* sentience, which could be a valuable tool for future research. This is because we would have both the "wet lab" DishBrain system, where *in vitro* neurons are interfaced with a game environment using an electrode

grid, and a low-cost and simple simulation that has its differences and similarities mapped. PingPOMDP simulates the interface used to communicate with the neurons, not the neurons themselves, so that researchers can focus on building the models that simulate the neuronal dynamics. Our work scope is to achieve comparable performance in the modified Pong game, but the same GridLink interface can be used for other experiments to build and compare models that exhibit the behavior of interest with *in vitro* neurons through the DishBrain system. We envision this project as a computational structure to help researchers experiment computationally and plan physical experiments with MEAs. The GridLink offers an easy abstraction to define the layout schematic and feedback protocols to interface the model with the environment.

The following sections are organized as follows. First, I will present the Pong environment and its specifications. Second, I will describe the DishBrain system and its components. Third, I will explain the proposed GridLink and its functions. Fourth, I will define the AIF model architecture and its parameters. Finally, I will introduce the PingPOMDP system and its objectives.

## 3.1 Pong Game Environment

The Pong environment is a modified version of the classic arcade game "Pong" as described by Kagan et al. (2022). This version has only one paddle. The game consists of hitting a ball with the paddle to prevent it from reaching the left wall. The game starts with the ball being thrown from the right side of the screen in a random direction. The game is updated with the ball moving at a constant speed and bouncing off the top, bottom, and right walls until it hits the edge of the play area behind the 'paddle', which marks the end of one 'rally' of pong. The player can control the paddle by moving it up or down. Each time the paddle hits the ball, the player scores a point. Figure 4 shows an example of the Pong game, where the ball moves towards the bottom left corner and the paddle moves down to intercept it.

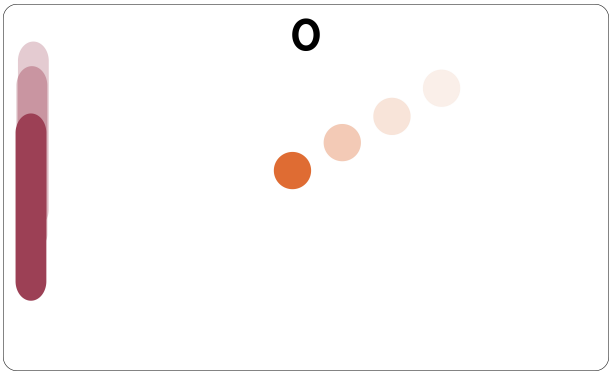


Figure 4 – Pong Game

The state of the game is defined by the coordinates of the ball and the paddle, as presented in Figure 5

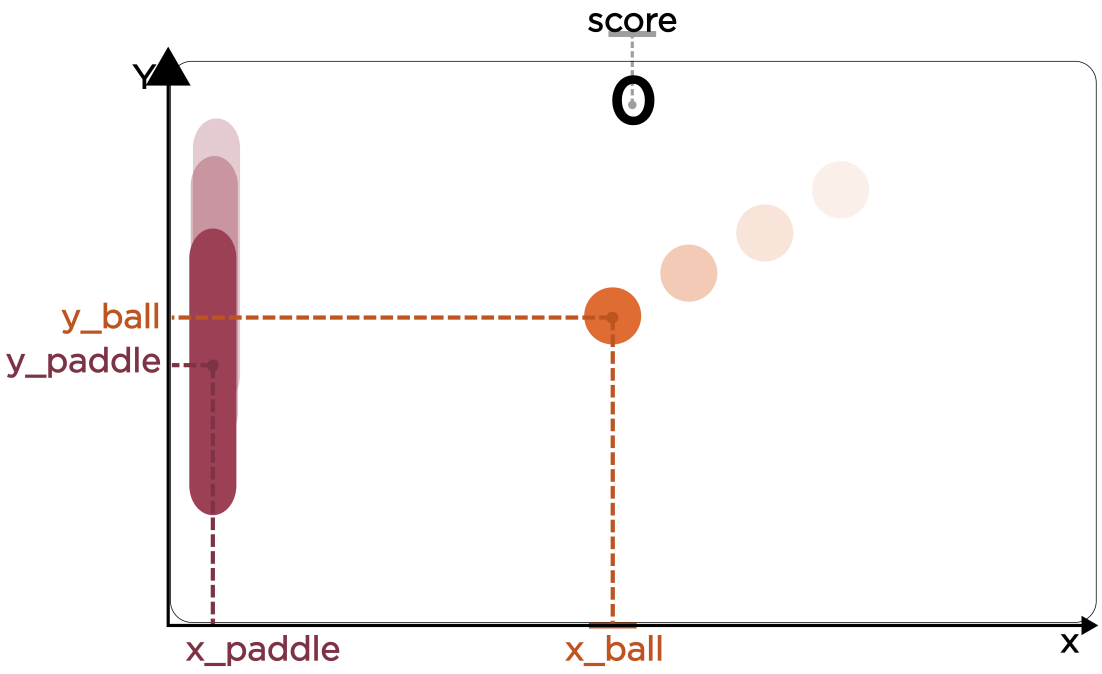


Figure 5 – Pong environment state is defined by the variables encoding ball and paddle position

The game controls are the commands to move the paddle up or move the paddle down. The game environment runs in steps where each frame advances the ball trajectory and updates the paddle position according to the received command.

## 3.2 DishBrain Electrode Grid

The DishBrain is the system developed by Kagan et al. (2022) to embed neuron cultures in the simulated Pong through a closed-loop of electrophysiological stimulation and recording via high-density multielectrode arrays (HD-MEAs) chips. This system encodes the ball position relative to the paddle (sensory stimulation) and reads arbitrarily defined regions of the culture to move the paddle up or down (if region 1 has more activity than region 2 the paddle moves up). Hits receive predictable feedback while misses receive unpredictable feedback. The rationale is that the cultures would "modify internal activity to avoid adopting states linked to unpredictable external stimulation. This minimization of input unpredictability would manifest as the goal-directed control of the simulated "paddle" in this simplified simulated "Pong" environment" (Kagan et al., 2022).

Figure 6 presents the electrode layout schematic used in their work. The MEA is divided into a "sensory area" and four "motor regions". The sensory area encodes relative

ball position and delivers predictable and unpredictable feedback accordingly. The motor regions reads the local neuronal activity to control the paddle.

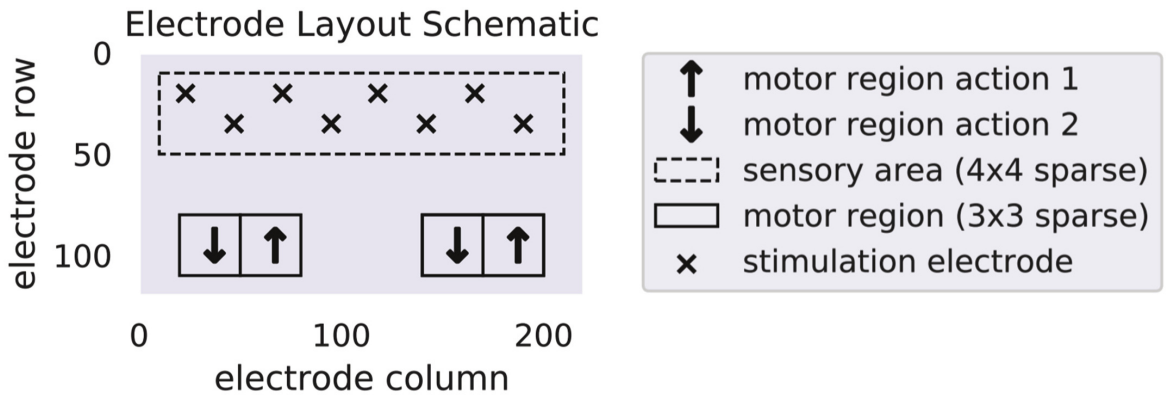


Figure 6 – DishBrain Electrode Layout Schematic (Kagan et al., 2022)

Following, we detail the mechanism of DishBrain's representation of the game environment according to the specifications in Kagan et al. (2022).

- The 'Sensory' area consists of 626 electrodes, where eight stimulation electrodes are embedded.
- During each rally the location of the ball relative to the paddle is encoded as stimulation to one of the eight stimulation electrodes, which is tracked in an internal 'stimulation sequencer' module. 75 mV was chosen as the sensory stimulation voltage, place coding was combined with a rate coding that delivered stimuli at 4 Hz when the ball was closest to the opposing wall and increased in a linear fashion to a max of 40 Hz as the ball reached the paddle wall.
- The remaining electrodes were divided into predefined motor regions on the MEA, consisting of four regions that were defined either as motor region 1 or motor region 2 as shown in Figure 6. This configuration was selected as it offered the possibility for biologically relevant features and minimized the chance of apparently successful performance through bias alone. Only activity in motor regions contributed towards paddle movement. The activity was measured over these two regions, where the region with higher activity would move the paddle in a corresponding direction. Activity in motor region 1 moved the paddle 'up' and activity in motor region 2 moved the paddle 'down'.
- Spikes are counted over a period of 10 milliseconds (200 samples), at which point the game environment is given the number of spikes detected in each of the configured electrodes in predefined motor regions as described above. These spike counts are

interpreted as motor activity depending on which motor region the spikes occurred in, thereby moving the 'paddle' up or down in the virtual space.

- At each of these 10ms intervals, the pong game is also updated, with a ball moving around a play area at a fixed speed 'bouncing' off the edges of the play area and off the paddle, until it hits the edge of the play area behind the 'paddle', which marks the end of one 'rally' of pong.
- Unpredictable stimulation was delivered to the cultures when a 'miss' occurred – i.e., when the culture failed to line the 'paddle' up to connect with the 'ball'. In order to add unpredictable external stimulus into the system, this feedback stimulus was set at 150 mV voltage and 5 Hz. This stimulation occurred at random sites at a random timescale over the 8 predefined input electrodes, for a period of four seconds, followed by a configurable rest period of four seconds where stimulation is paused, followed then by the next rally.
- Predictable stimulation was delivered to cultures when a 'hit' occurred – i.e., when the cultures successfully lined up the 'paddle' to connect with the 'ball. This was delivered at 75mV at 100Hz over 100ms. This occurred at the instant when the simulated ball impacted the paddle and replaced other sensory information for the 100ms period. Predictable stimulation occurred at this frequency and period across all 8 stimulation electrodes simultaneously.

## 3.3 GridLink

GridLink is the proposed computational abstraction to the electrode grid in the DishBrain system. GridLink has a "grid" to represent the state of the "electrode grid", to represent some environment to the model, GridLink allows for the specification of its layout, behavior and feedback protocols. For clarity and scope of this project, I'll focus on following Kagan et al. (2022) specifications with the necessary abstractions. In this case, the grid mimics the layout schematic of DishBrain presented at seção 3.2. As the diagram in Figure 7 shows, at first I'll experiment with a very simple 8 by 4 matrix where each cell will have only two possible states corresponding to "on" or "off". This resolution is the smallest that can meaningfully analogize the DishBrain layout and protocols:

- Each cell of the grid represents an area of electrodes in DishBrain's electrode grid. The value of each cell (0 or 1) corresponds to whether there is electrical activity in that area.
- The grid has a "sensory area" with eight cells representing the stimulation electrodes in DishBrain. Those eight cells will encode the feedback protocols by changing their states from "0" to "1". Those protocols are presented at subseção 3.3.1.

- The "motor region" is also composed of eight cells. Divided into two pairs of motor regions. GridLink will map the model actions to this motor area, and also map the values in this area to actions in the environment, this process is presented at subseção 3.3.3
- The other cells remain in zero state all the time. The Figure 7 shows the minimum and maximum activation of the grid following this layout.

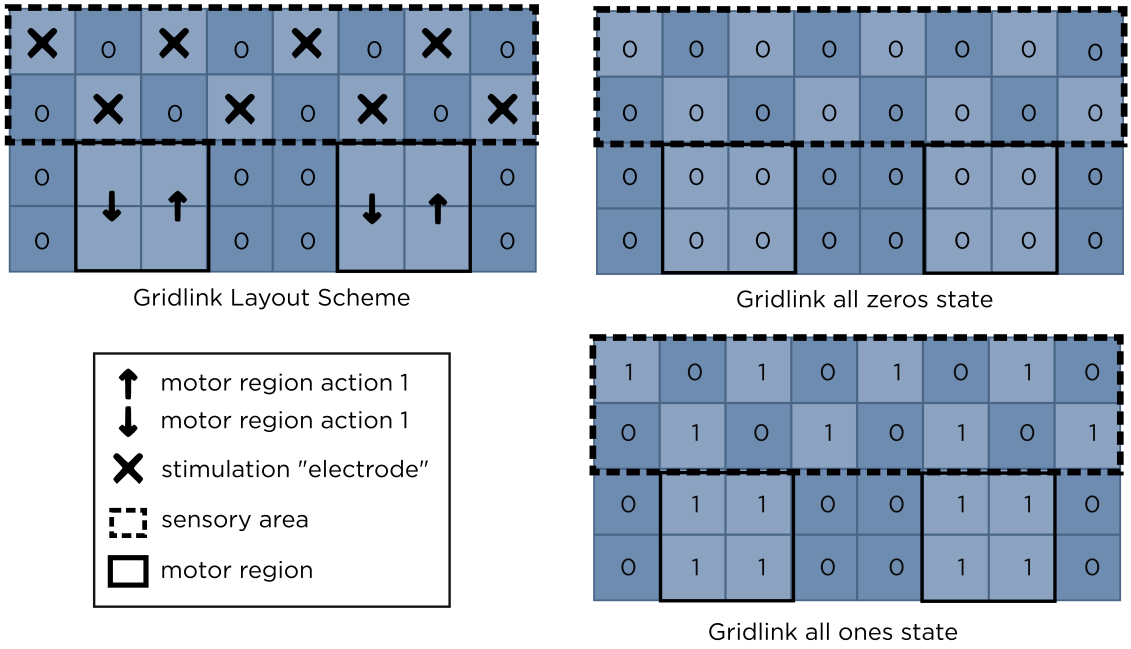


Figure 7 – GridLink layout in PingPOMDP

## 3.3.1 Feedback Protocols

Following the DishBrain specifications, there are three feedback protocols: sensory feedback representing the environment state, predictable feedback when the paddle hit the ball, and unpredictable feedback when the ball passes the paddle ending the rally.

### 3.3.1.1 Sensory Feedback

This feedback represents the pong environment state, it encodes the ball position in relation to the paddle by activating on of the eight stimulation cells. The highest ball in relation to the paddle will activate the cell further right, the lowest ball will activate the leftmost stimulation cell, and so forth. The Figure 8 presents the environment variables of interest (y_ball and y_paddle), their difference is mapped to one of the stimulation cells, represented in lighter blue. The relative_position variable is calculated via $relative\_position = y\_ball - y\_paddle$.

To illustrate this process the image used a vertically centralized paddle and a rotated presentation of the sensory area with a dotted line highlighting the activated cell. We can see the ball both above the paddle center and below the paddle center with the corresponding state of the sensory area.

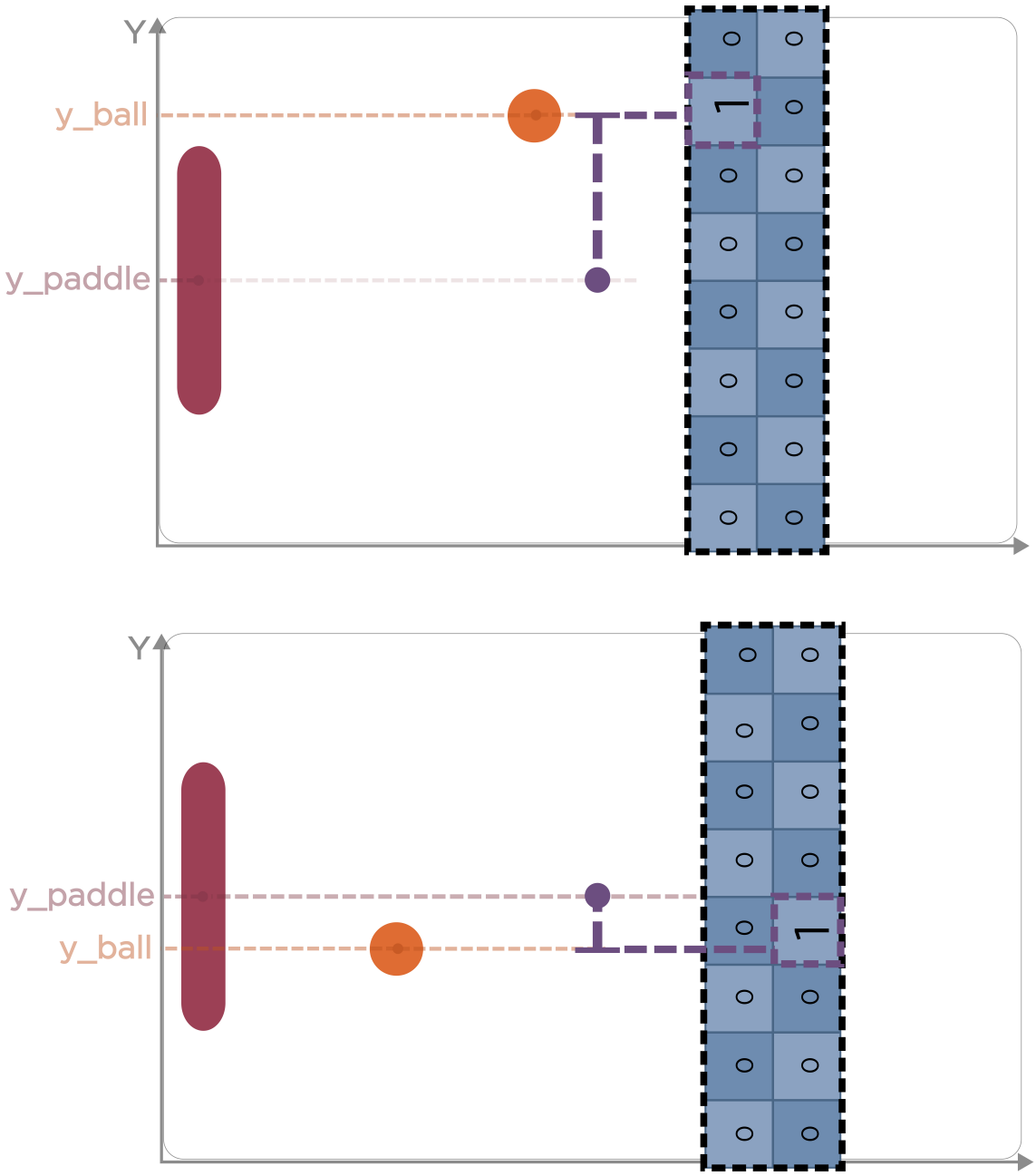


Figure 8 – Grid Sensory Feedback Protocol

#### 3.3.1.2 Predictable Feedback

The predictable feedback consists of a controlled sequence of activation and deactivation of all the stimulation cells simultaneously. This protocol is administered every time the paddle hits the ball.

The Figure 9 presents the sequence of environment states and sensory area states in the moments before (at lower opacity) and after the hit.

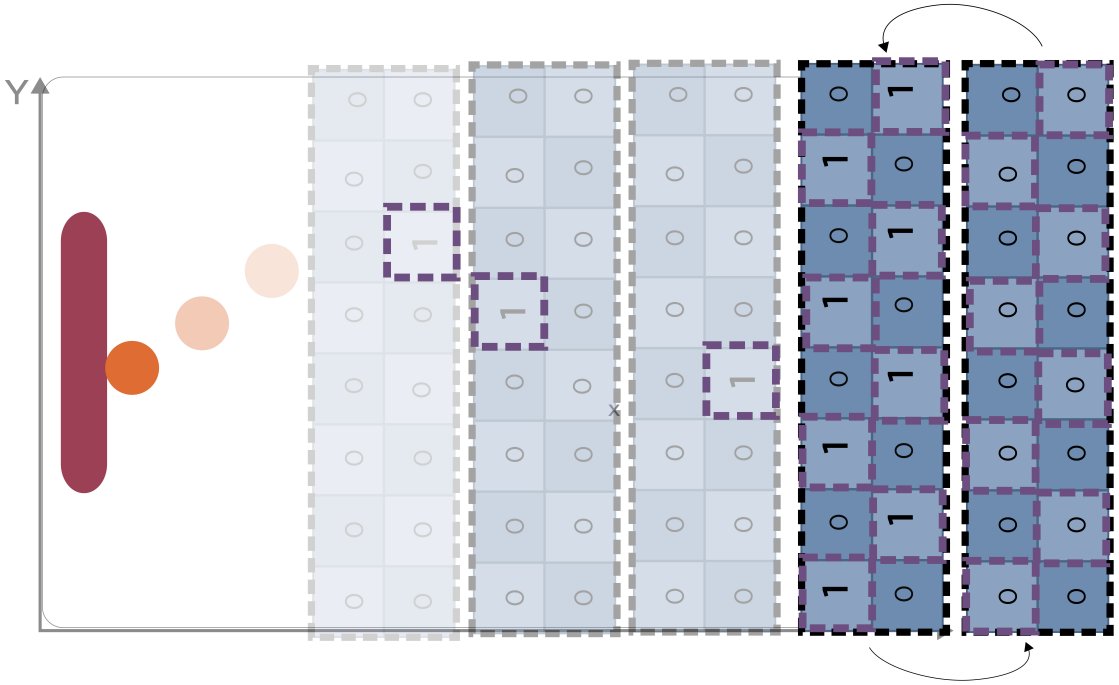


Figure 9 – Grid Sensory Feedback Protocol

### 3.3.2 Unpredictable Feedback

The unpredictable feedback consists of a sequence of random activation and deactivation of random stimulation cells. This protocol is administered every time the ball passes the paddle ending the rally.

### 3.3.3 Action Mapping

The grid represents actions in the eight cells corresponding to the "motor region" of the DishBrain. GridLink will map the model actions to this motor area, and also map the values in this area to actions in the Pong environment.

Figure 10 illustrate those eight cells by naming them C0 to C7. Those cells are arbitrarily considered to represent the actions of moving the paddle up or moving the paddle down. That is, C0, c4, c2 and c6 are the regions representing "move paddle down", while c1, c5, c3, and c7 represent "move the paddle up". Therefore, the region with more activated cells (the sum of their values) determines the action to be sent to the Pong environment.

The rationale of this encoding in DishBrain is that it provides interesting biological information about neuronal activity, and enforces the "intentionality" of the action by

minimizing the chance of successful performance through bias alone (Kagan et al., 2022).

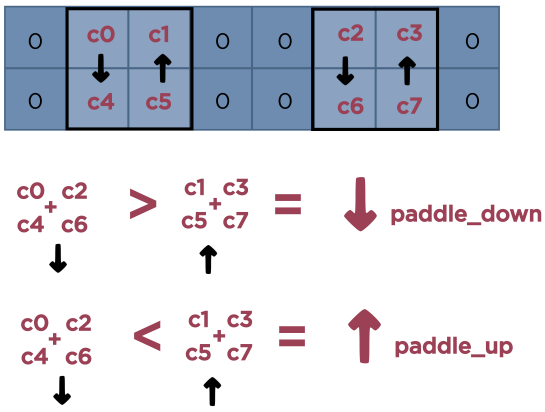


Figure 10 – Grid Action Encoding

## 3.4 AIF Model

PingPOMDP uses an AIF model, also called an agent, to interact with the Pong environment through the GridLink interface. As such, the agent receives observations from and acts on the Pong environment through the GridLink in a loop. This agent will be implemented using the pymdp library as presented in section 2.2.1, It is important to recap that:

- The agent has hidden states to encode its beliefs about the environment each possible observation has a probabilistic relation to each hidden state. Those likelihoods are encoded in the A matrix.
- The agent also has control states that are related to the hidden states by the B matrix.
- Optionally the agent can have a C vector to encode its preference for each possible observation.

Those are the main model parameters we will experiment with to establish learning and adaptive behaviour in the PingPOMDP system.

### 3.4.1 pymdp Agent

The Listings 3.4.1 show an example of how to use pymdp to build and run an active inference process. First, we define the number of possible factors for each modality of observation and state (hidden and control). In this example, we have two modalities of observation with 3 and 5 possible sensory outcomes each. We also have three hidden

states with 4, 2, and 3 possible values for each latent variable that the agent believes can generate the observations. Finally, we have three control states with 4, 1, and 1 possible actions each.

Next, we create the **A** and **B** arrays as uniform distributions in the shape of their components (observations over states, and transitions from states considering each possible action). We also define a uniform **C** vector and instantiate an agent using **A**, **B**, and **C**.

Finally, we execute AIF by giving the agent a random observation as a list specifying the indices of the observation for each observation modality. The agent then updates its posterior over hidden states, infers policies, and samples an action. The agent is then ready to begin the cycle again with a new observation.

```
import pymdp
from pymdp import utils
from pymdp.agent import Agent

num_obs = [3, 5] # observation modality dimensions
num_states = [4, 2, 3] # hidden state factor dimensions
num_controls = [4, 1, 1] # control state factor dimensions
A_array = utils.random_A_matrix(num_obs, num_states) # create sensory
    likelihood (A matrix)
B_array = utils.random_B_matrix(num_states, num_controls) # create
    transition likelihood (B matrix)

C_vector = utils.obj_array_uniform(num_obs) # uniform preferences

# instantiate a quick agent using your A, B and C arrays
my_agent = Agent( A = A_array, B = B_array, C = C_vector)

# give the agent a random observation and get the optimized posterior
    beliefs

observation = [1, 4] # a list specifying the indices of the observation,
     for each observation modality

qs = my_agent.infer_states(observation) # get posterior over hidden
    states (a multi-factor belief)

# Do active inference

q_pi, neg_efe = my_agent.infer_policies() # return the policy posterior
    and return (negative) expected free energies of each policy as well

action = my_agent.sample_action() # sample an action from the posterior
    over policies
```

## 3.5 PingPOMDP

PingPOMDP is the system that orchestrates the reciprocal cycles between the agent (AIF model) and the Pong environment through GridLink. As the Figure 11 presents, at each time step:

1. The game takes one step updating the ball and paddle positions and returning the environment state composed of the positions and if there is an event (hit or miss).
2. If there is a hit or miss, the system administers the predictable or the unpredictable protocol to the agent. Else, the system provides sensory feedback to the agent.
3. The agent receives the observation, updates its internal representations, and takes action.
4. The cycle repeats

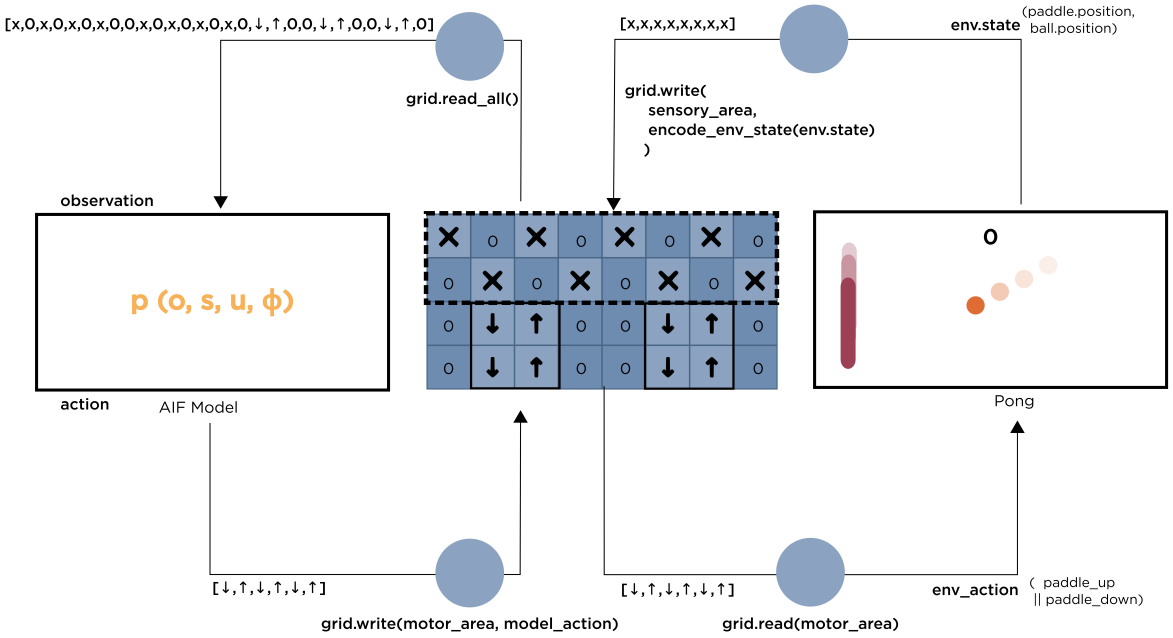


Figure 11 – PingPOMDP conceptual diagram

The Listings 3.5 presents an abstract pseudocode for this dynamics.

```
1 import gridlink
2 import pong
3 import model
4
5 class PongGridlink(gridlink.Gridlink):
6     def observe(self):
7         pass
```

```
8
9       def act(self):
10          pass
11
12  class PingPOMDP:
13      def __init__(self):
14          self.pong = pong.Pong()
15          self.grid = PongGridlink()
16          self.model = model.Model()
17
18      def predictable_feedback(self):
19          pass
20
21      def unpredictable_feedback(self):
22          pass
23
24      def agent_observe_and_act(self, observation): #-> action:
25          pass
26
27      def step():
28          if self.pong.event == "hit!":
29              self.predictable_feedback()
30          elif self.pong.event == "miss":
31              self.unpredictable_feedback()
32          else:
33              observation = grid.observe(pong.state)
34              action = agent_observe_and_act(observation)
35              pong.step(grid.act(action))
```

# References

Adams, R. A.; Vincent, P.; Benrimoh, D.; Friston, K. J.; Parr, T. Everything is connected: Inference and attractors in delusions. *Schizophrenia Research*, v. 245, p. 5–22, Jul 2022. ISSN 0920-9964. Available on: <https://www.sciencedirect.com/science/article/pii/S0920996421003054>.

Baltieri, M.; Buckley, C. L. Pid control as a process of active inference with linear generative models. *Entropy*, Multidisciplinary Digital Publishing Institute, v. 21, n. 33, p. 257, Mar 2019. ISSN 1099-4300. Available on: <https://www.mdpi.com/1099-4300/21/3/257>.

Costa, L. D. et al. Active inference on discrete state-spaces: A synthesis. *Journal of Mathematical Psychology*, v. 99, p. 102447, Dec 2020. ISSN 0022-2496. Available on: <https://www.sciencedirect.com/science/article/pii/S0022249620300857>.

Heins, C. et al. pymdp: A python library for active inference in discrete state spaces. *Journal of Open Source Software*, v. 7, n. 73, p. 4098, May 2022. ISSN 2475-9066. ArXiv:2201.03904 [cs, q-bio]. Available on: <http://arxiv.org/abs/2201.03904>.

Heins, C. et al. pymdp: A python library for active inference indiscrete state spaces. *Journal of Open Source Software*, v. 7, n. 73, p. 4098, May 2022. ISSN 2475-9066. Available on: <https://joss.theoj.org/papers/10.21105/joss.04098>.

Kagan, B. J. et al. In vitro neurons learn and exhibit sentience when embodied in a simulated game-world. *Neuron*, p. S0896627322008066, Oct 2022. ISSN 08966273. Available on: <https://linkinghub.elsevier.com/retrieve/pii/S0896627322008066>.

Parr, T.; Pezzulo, G.; Friston, K. J. *Active Inference: The Free Energy Principle in Mind, Brain, and Behavior*. The MIT Press, 2022. ISBN 9780262369978. Available on: <https://doi.org/10.7551/mitpress/12441.001.0001>.

Smith, R.; Badcock, P.; Friston, K. J. Recent advances in the application of predictive coding and active inference models within clinical neuroscience. *Psychiatry and Clinical Neurosciences*, v. 75, n. 1, p. 3–13, 2021. ISSN 1440-1819. Available on: <https://onlinelibrary.wiley.com/doi/abs/10.1111/pcn.13138>.

Smith, R.; Friston, K.; Whyte, C. A step-by-step tutorial on active inference and its application to empirical data. PsyArXiv, Jan 2021. Available on: <https://psyarxiv.com/b4jm6/>.

Tschantz, A.; Baltieri, M.; Seth, A. K.; Buckley, C. L. Scaling active inference. In: *2020 International Joint Conference on Neural Networks (IJCNN)*. 2020. p. 1–8. ISSN 2161-4407.