Analyzing the Computational Capabilities of a Candidate Biological Organism

Christian Pratt

Department of Physics and Astronomy
University of California-Davis
czpratt@ucdavis.edu

June 9, 2022

Abstract

Physical systems which exhibit computational properties and competencies are especially prevalent in the modern world. Traditionally, classical and quantum computers take center stage as to what a computer is defined to be. An alternative approach to defining what it means for a system to compute is to study unconventional candidates of a computer: A substrate which receives information from its environment, processes the information, produces some output, and is capable of being programmed. A minimal biological organism, C. elegans, is considered to be a computational substrate, first by theoretically deconstructing the organism into its fundamental constituents, and then experimentally conducting simulations based on certain iterative criteria. From these implementations, it is clear that biological organisms receive and process information from its environment, and produce output in the form of movement. The notion of programmability is discussed, but not experimentally validated.

I. Introduction

Motivation

The association between physical systems and their computability is not commonly discussed in traditional physics, yet the association's prevalence can be demonstrated in many contexts. Classical and quantum computers have taken a central role as being canonical computing devices, known for their ranges of multi-purposeful utility for solving various classes of problems. Alternative forms of computing have begun to achieve greater prominence, such as chemical computing [8]. That being said, currently the best framework which understands physical systems and their properties is the human brain. Because humans are biological organisms, their ability to comprehend, predict and analyze physical system's characteristics can out compete other computing substrates in diverse aspects.[1] It is therefore necessary to consider the computability of a biological organism in the minimal case, and to analyze whether this organism exhibits computational properties or not. In this project, the minimal model, which is also analyzed in many other fields, *C. elegans*, is utilized to study the computational properties of a simplified adaptation of *C. elegans*. The goal is to demonstrate that *C. elegans* embodies specific qualities such that it can be considered a substrate which computes. This topic is of interest because the conventional association of "computability" is generally associated with the classical and quantum platforms of computation. If a framework is constructed such that biological organisms can also exhibit computational properties, but in different ways than classical and quantum computing devices, this gives rise to the notion that computability is independent of substrate, although the classes of problems which can be solved will vary between substrates, as visualized in Figure (1).

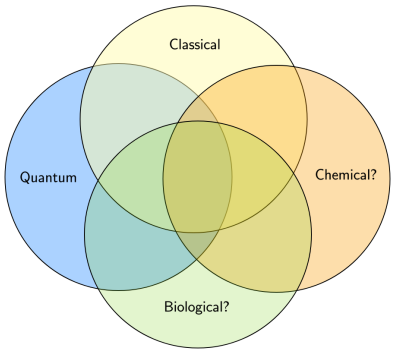

Figure 1: *Classes of problems which are solved by a particular computing substrate. Because chemical and biological computing aren't completely solidified as computing platforms, they are respectively indicated with a '?'. Regions of solvable classes of problems are not to scale.*

Synopsis of project and results

In order to investigate the computational capabilities of a candidate biological organism, a minimal example must be selected. One organism which satisfies elementary criteria, such as only being equipped with a basic set of features, the organism's movement is uniquely described by a small set of components, etc. is the nematode *C. elegans*. Next, the computational properties of the candidate biological organism *C. elegans* are analyzed by first breaking *C. elegans* down to fundamental simplified constituents, whose conglomeration will be known as an *agent*. The agent will be studied by attempting to exhibit behaviors via different iterations of an experiment. By enabling the agent to conduct tasks in this experiment, respective analyses and conclusions can be reached.

Utilizing the agent simulation software `NetLogo` [11], initial results, in which an experimental simulated agent whose internal capabilities are increased, with regards to the agent's decision making for navigating its environment, are displayed and discussed.

By modifying how the agent is able to move in the environment, and allowing the agent to achieve a type of task completion, such as searching for a reward, will demonstrate different outcomes. One possible outcome is if an agent, equipped with a memory for what a reward looks like and how to distinguish environmental differences to help achieve a task, vs. if the agent had no memory, would show that possessing a memory would greatly aid the agent in its mission. These results, however, don't demonstrate learning, but rather it can be inferred that the agent is capable of carrying out

[1] Classical and quantum computers are dominant for carrying out calculations. Despite that, these computing substrates are not as good as humans at the discussed tasks.

pre-programmed tasks, such that the agent was equipped with the capabilities to execute the task before the experiment began. More experimentation would be required to carry out an implementation of an agent which learns.

II. Background

To begin, a brief introduction on the traditional prevalence of computing will be provided, in order to lay the groundwork for how other platforms of computing would provide importance to society at large. First, classical computing, which, in modern times, are extremely prevalent. Presenting some statistics for classical computing devices, current projections show that there will be roughly 18 billion mobile devices by 2025 [9], and as of 2016 there were roughly 2 billion laptop users [7], and at least tens of thousands of supercomputers exist today. Classical computing is everywhere, and their utility continues to grow as time goes on.

However, quantum computing is another realm of computation which is being readily developed by the likes of many companies including Google [2], IBM [5], D-WAVE [4], among others, as well as many university research institutions. The paradigm of quantum computers can be broken up into three categories: analog, annealing and universal, but will not be discussed further.

As mentioned previously, chemical computers are now being researched at institutions [8] with the hopes of being able to synthesize any molecule, which would have many kickbacks to society if continuing success is upheld.

Building on these foundations, an overview of the essential criteria for what makes a substrate a computer will be undertaken. Essentially the tasks that a "computer" carrys out, are the following:

- ✧ Reads environmental input for:
  - ✦ Processing (short term memory)
  - ✦ Storage (long term memory)
  - ⇒ Both are interconnected (*processed input*)
- ✧ Processed input is utilized for output:
  - ✦ Observed decisions (calculations, movement, etc.)
- ✧ Being *programmable* (able to be modified)
  - ✦ e.g. classical computer programs are modified via code by the programmer.

where the first two main items can be more readily understood across different candidate computing substrates, while the third item can be more complicated. For classical and quantum computers, if the environment is the keystrokes generated by the programmer, then the computer is able to be modified at the will of the environment. With a less conventional computing substrate, such as a biological organism, and along a similar line of reasoning, this means that the organism must adapt according to its environment, i.e. the organism must *learn* in order to adapt to its environment. This notion will be discussed in the **Results** section (V).

Because, for example the human brain (and thereby part of a biological organism) is the best known tool for understanding the known laws of physics, it would be ideal to know how the human brain can receive, process, store and output information. Nevertheless, this would be too vast of a problem to undergo in this project, so it is desirable to consider a fundamental example of a biological organism which demonstrates computational capabilities. One organism in particular, which is studied across various fields including biology, neuroscience, etc., is *Caenorhabdidtis (C.) elegans*, which is a small nematode or roundworm of 1mm in length [12]. There are many structural properties of *C. elegans*, but what motivates this research project are the following characteristics:

- ✧ The entire neural network of *C. elegans* is completely mapped and visualized [3].
- ✧ Stochastic behavior is exhibited by a few parameters describing the structure of *C. elegans* [6].
- ✧ Accessible enough of a subject to conduct simplified simulations.

Due to these benchmarks, *C. elegans* is an optimal candidate for asking the question: *Is a biological organism a computer?*

Elements of the essential anatomy of the *C. elegans* is presented in Figure (2). The organs of concern for this project are ones which receive and process energy or sensory input from the environment. The organs which conduct these duties are the *pharynx* and *head*, and the organs which process this energy

or sensory input are the *intestine* and *ventral nerve cord*, respectively.

Figure 2: *Anatomy of C. elegans, which includes the discussed essential anatomy essential for modelling the organism as an agent.*

Effectively, the *C. elegans* will receive some environmental input (light, heat, etc.) and demonstrate some output, which is shown by movement corresponding to the organism's decisions. This will important for the **Methods** section (IV).

III. Dynamical System

The dynamical system to be studied in this project is the *agent*, which is a simulated construction of *C. elegans*, and will be detailed in the **Methods** section (IV). As mentioned previously, the agent would exhibit stochastic behavior, even though it is a minimal organism [6]. How the agent moves throughout its environment is dictated by the agent's properties regarding how to navigate itself to satisfy a certain task, or carry out a specific behavior, which takes the place of *memory*. This is because, if the agent is instantiated with the ability to discern internal environmental differences, this suggests that the agent can readily adapt its decisions based on the task at hand. Therefore, the more expressed criteria available to the agent in the initialization stage, the more able the agent will be to achieve task completion.

Equations of motion

In terms of the eigenworm basis described by [10], to support the stochasticity and random behavior that the *C. elegans* movement exhibits, in terms of the phase angle $\phi(t)$ and phase velocity $\omega(t)$, the *C. elegans* activity is described by Brownian motion, utilizing the Langevin equation:

$$\begin{aligned} \frac{\mathrm{d}\phi(t)}{\mathrm{d}t} &= \omega(t) \\ \frac{\mathrm{d}\omega(t)}{\mathrm{d}t} &= \mathrm{F}(\phi(t),\omega(t)) + \eta(t)\sigma(\phi(t),\omega(t)) \end{aligned} \tag{1}$$

where $\mathrm{F}(\phi(t),\omega(t))$ is the average acceleration in terms of phase and phase velocity, and $\eta(t)$ defines a random noise function which will have some correlation to the change in phase velocity over time, $\sigma(\phi(t),\omega(t))$. [10]. Utilizing the first two eigenworms in the eigenworm basis, Figure (3) demonstrates how the eigenworms affect the movement of the *C. elegans* itself.

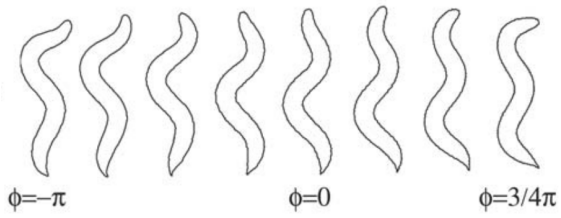

Figure 3: *The variation in phase angle in terms of the first two eigenworms, serving as a visualization tool for the Langevin equation. From [10]*

When not taking the eigenworm basis into account, the dynamical behavior of *C. elegans* can be captured by the canonical Hamilton equations of motion, described as a coupled system of damped, driven Hamiltonian oscillators,

$$\begin{aligned} \dot{Q}_i &= \frac{\partial \mathrm{H}}{\partial P_i} \\ \dot{P}_i &= -\frac{\partial \mathrm{H}}{\partial Q_i} + \mathcal{C}(Q_i, P_i, \psi(t)) \end{aligned} \tag{2}$$

where H is the Hamiltonian describing the dynamical and elastic behavior of *C. elegans*, (Q_i, P_i) are the generalized position and momentum coordinates corresponding to the ith normal mode of the system, and $\psi(t)$ is a function which encompasses the proprioceptive feedback and neural processing of the *C. elegans* coming into contact with the environment [1].

IV. Methods

First, a theoretical approach for generating an agent in terms of fundamental components will be taken. Then, an experimental technique describing how such an agent would be simulated will be presented.

Theoretical Implementation

Outlining the actions of a simplified version of *C. elegans*, it will detect input from its environment, in the form of light and heat, and demonstrate some output in the form of decision-based movement. Concisely, this can be formulated in terms of a *configuration alphabet*:

$$\begin{aligned} A_{\text{config}} &= \{\text{read environment}, \text{control decisions}\} \\ &= \{\mathrm{S}_{\text{env}}, \mathrm{S}_{\text{dec}}\} \end{aligned} \tag{3}$$

where $\mathrm{S}_{\text{env}} = \{\text{photoelectric}, \text{temperature}\}$ and $\mathrm{S}_{\text{dec}} = \{\text{actuators}\}$. The composition of these sensors will now formally comprise an *agent*, a term which has been frequently used previously, but is now grounded with some formalism. The term, agent, is how *C. elegans* will be referred to for the remainder of this paper.

Now, consider how the photoelectric sensor would theoretically function within its environment for some constant temperature T, so the actuators are solely receiving input from the photoelectric sensor. Suppose the sensor can only read the colors `Red (R)` and `Green (G)` from the environment for simplicity, which is summarized in Table (1):

Environmental input	R	G
Output to actuators	0	1

Table 1: *Photoelectric sensor relays the detected environmental input,*

which can be formulated as the function $f : \mathcal{A}_{\text{color}} \mapsto \mathcal{A}_{\text{info}}$ such that $\mathcal{A}_{\text{color}} \equiv \{\mathtt{R}, \mathtt{G}\}$ and $\mathcal{A}_{\text{info}} = \{0, 1\}$, implying that $f(\mathtt{R}) = 0$, and $f(\mathtt{G}) = 1$. Then, when the agent relays this information to the actuators, this can be defined as a function $g : \mathcal{A}_{\text{info}} \mapsto \mathcal{A}_{\text{output}}$, where $\mathcal{A}_{\text{output}}$ corresponds to the agent's *shape basis*, which is the set of elements corresponding to the agent's demonstrated actions within the environment. For example, the shape basis can be formulated as $\mathcal{A}_{\text{output}} = \{\mathtt{forward}, \mathtt{right}, \mathtt{halt}\}$ where it is implied that $\mathtt{reverse} = (\mathtt{forward})^{-1}$, $\mathtt{left} = (\mathtt{right})^{-1}$, while `halt` has no "inverse". Then the function g can be assigned as $g(0) = g(f(\mathtt{R})) = \{\mathtt{halt}\}$, whereas $g(1) = g(f(\mathtt{G})) = \{\mathtt{forward}, \mathtt{right}\}$. Furthermore, the ideal agent would follow these assignments with probability 1, but it is more realistic to have the agent's actions be dictated in a probabilistic manner, e.g.
$\Pr(\mathtt{halt}|g(0)) = \varepsilon \Rightarrow \Pr(\mathtt{halt}|g(1)) = 1 - \varepsilon$.

Recapping, the photoelectric sensor is formulated in terms of binary information processing for simplicity at a constant temperature. As a succinct remark, the temperature sensor would be constructed in an analagous manner, except the environmental input would appear to be some $T < \tilde{T}$ or $T > \tilde{T}$, where $\tilde{T}$ is a criterion for some action. This concludes the discussion of the theoretical implementation of the agent.

Experimental Implementation

For implementing a simulation of the agent, *NetLogo*, an agent-based modelling software system useful for the efficient modelling of proxies [11], was employed.

There are almost limitless possible experiments one could assemble to demonstrate how the agent advances through its environment. One such possibility involves the agent, equipped with a "photoelectric sensor" which is able to detect two colors; green and red. Suppose that the color red is associated with a form of "halt", such that once the agent finds the color red, that red color is moved to another location within the environment, while every other portion of the environment is the color green. Then if the agent discovers the color green, the agent will continue to move throughout the environment until it reaches the color red. The red color would be transported to another location, and the process would repeat for any number of trials, simply visualized in Figure (4).

Two iterations of the experiment will now be discussed. The agent's pre-programmed criteria will be modified in both experiments, such that the first iteration includes an agent whose movements are random, while the second iteration consists of an agent whose movements are decisive but with the enforcement of slight random perturbations to its movement. The simulated environment is composed of a grid of 50×50 "patches" (grid squares), bounded by an outer square. Time in `NetLogo` can be measured discretely in the units of *ticks*, which will be exercised throughout this experimental application.

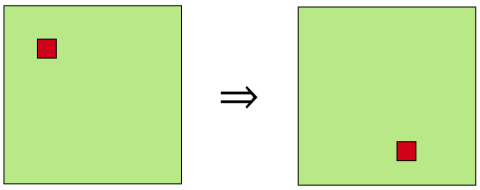

Figure 4: *Proposed experiment demonstrating how the agent would make decisions within its environment. Once the agent finds the red color, that red color is moved to another location within the environment, and the agent is tasked with finding it once more.*

V. Results

Experiment 1, Iteration 1

The agent is only able to move in a random direction, emulating a non-existent memory of its past moves and its goals. The agent will continuously move in a random direction provided by randomized (x, y) coordinates. A generic algorithm for the agents behavior is given by

Algorithm 1 Agent Behavior, Implementation 1

Require: $x_{\text{red}} = \texttt{random float} : -50 \leq n \leq 50$
Require: $y_{\text{red}} = \texttt{random float} : -50 \leq n \leq 50$
Require: $x_{\text{agent}} = \texttt{random float} : -50 \leq n \leq 50$
Require: $y_{\text{agent}} = \texttt{random float} : -50 \leq n \leq 50$

while $t < t_{\text{max}}$ **do**
<Agent rotates $0 \leq$ degrees ≤ 360>
<Agent moves forward one patch>
if current patch == red patch **then**
<Red patch is randomized to another (x, y) coordinate>
else
<pass>
end if
end while

This algorithm is demonstrated by the Figures (5) and (6).

The amount of "ticks" that it would take for the agent to locate the red square can be visualized with a histogram, demonstrating the randomness of the agent's behavior in Figures (7) and (8), where the latter histogram focuses the readers' attention to the plot structure where the majority of captures are recorded.

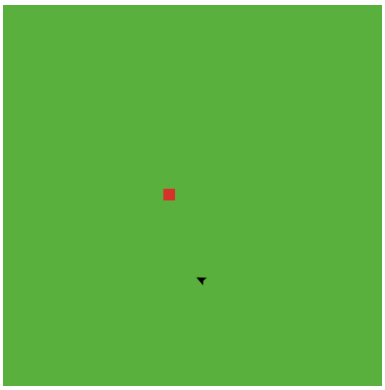

Figure 5: *Agent before "capturing" red square, serving as the goal of the agent.*

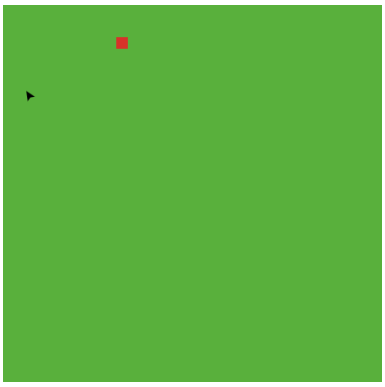

Figure 6: *Agent after "capturing" red square, and red square moves to another location for the agent to find.*

Due to the agent having no "memory", rather lacking the sensory power of determining the distinguishing colors, this forbade the agent to locate the red square in a timely manner, as demonstrated by the plots. The mean time it took for the agent to find the red square was ≈ 2271, which is excessively long. This is due to the random nature of the agent, consequently due to the agent lacking a memory of its surroundings and itself. If the agent were able to have "memory" of how to discern colors, it can be readily concluded that the agent should not take much time to find the red square, which is precisely Iteration 2 of this experiment.

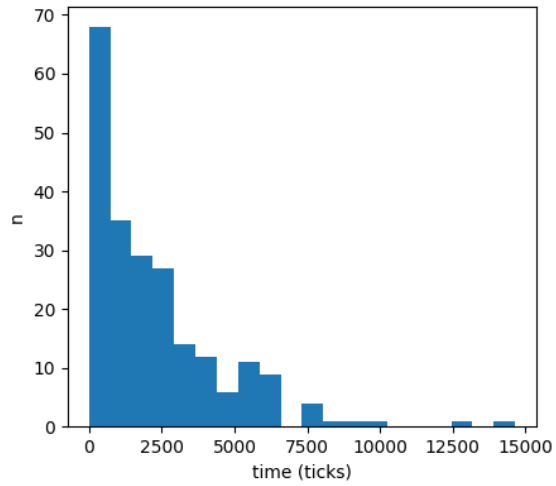

Figure 7: *Histogram of displaying the number of ticks the agent took to find the red square. Total ticks = 500, 000, number of trials = 220, number of bins = 20, mean ≈ 2271*

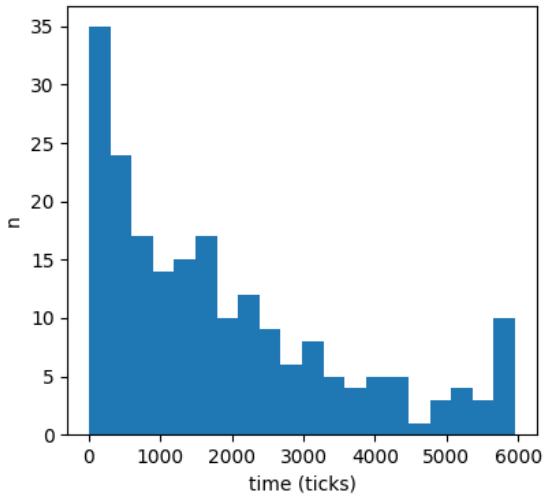

Figure 8: *A more focused version of Figure (7).*

Experiment 1, Iteration 2

The agent is now equipped with the adeptness of detecting where the red square is located within the environment, although some randomness is introduced into the movement of the agent to obtain more sensible results. A nonexclusive algorithm for accomplishing this task is given by Algorithm (2), which demonstrates the discussed ideology of the experiment.

Algorithm 2 Agent Behavior, Implementation 2

Require: $x_{\text{red}} = \texttt{random float} : -50 \leq n \leq 50$
Require: $y_{\text{red}} = \texttt{random float} : -50 \leq n \leq 50$
Require: $x_{\text{agent}} = \texttt{random float} : -50 \leq n \leq 50$
Require: $y_{\text{agent}} = \texttt{random float} : -50 \leq n \leq 50$

while $t < t_{\text{max}}$ **do**
<Agent rotates $0 \leq$ degrees ≤ 360>

if (agent > distance to red square - arbitary distance) **then**
<Agent faces towards red square>
<Agent perturbed by small random rotation away from red square>

else
<pass>
end if
<Agent moves forward one patch>
if current patch == red patch **then**
<Red patch is randomized to another (x, y) coordinate>
else
<pass>
end if
end while

In Figures (9) and (10), data from the simulated experiment is shown, which follows the same framework as in the first iteration.

Rather interestingly, even though the agent's movement was not randomized in Iteration 2, the outcome of how long it took for the agent to find the target forms a Gaussian-esque distribution. This could be due to the randomization of where the target was placed after the agent found it, but not enough experiments were completed to solidify this conclusion. The agent undoubtedly did not take as

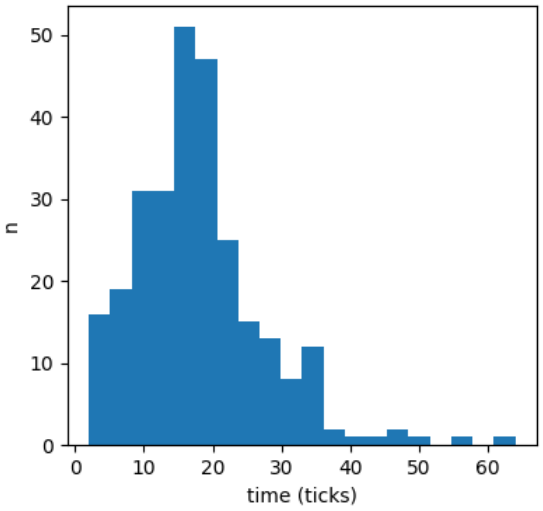

Figure 9: *Iteration* 2 *of Experiment* 1. *Total ticks* = 5000, *number of trials* = 277, *number of bins* = 20, *mean* ≈ 18.

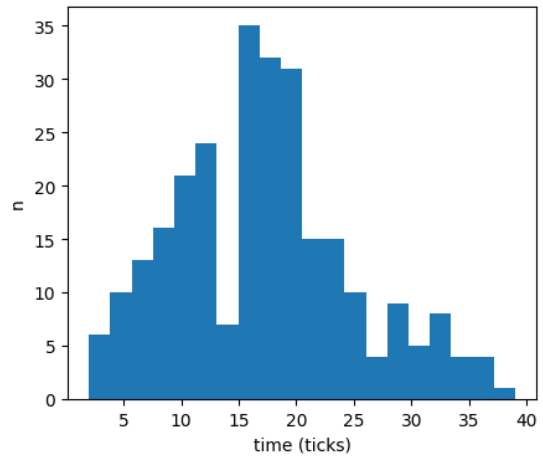

Figure 10: *A more focused version of Figure (9).*

long to find the target as much as the agent did in Iteration 1, with a mean of only ≈ 18 ticks.

Future work

These experiments could be improved in many ways. Firstly, a larger quantity of varied experiments can be conducted, with similar iterations as the ones mentioned here, primarily with the goal of demonstrating an agent learning, i.e. the agent has to probabilistically adapt to its environment to achieve some task. Secondly, more statistical quantities can be computed, such as standard deviations, etc. Lastly, this foundation would ideally be studied in order to relate concepts of computational mechanics to the experimental outcomes as well. Perhaps the theoretical implementation of the agent would be more ideally suited for such an undertaking, where two functions could be created which optimize for *survival* and *reward*. The agent readily employs these functions as it traverses the environment, especially during the intermediate input processing stage of data being sent to the actuators. However, these functions would optimize for some *probabilistic potential*, due to organisms not learning from errors each time they make them. This is concisely indicated as

$$\text{Input} \to \text{Process} \to \text{Output}$$

vs.

$$\text{Input} \to \text{Process} \to \text{Optimization} \to \text{Output}$$

where the former case is demonstrated in this paper. Then, suppose if the agent associates *causality* with these functions, such as if, after a non-zero time in its environment, the agent associates the causal states with its environment:

- ✧ $\mathcal{S}_0 \mapsto$ (halt in red zone to obtain food)
- ✧ $\mathcal{S}_1 \mapsto$ (move in green zone to search for food)

then the agent has developed *learning*, i.e. the agent actively optimized its probabilistic potential for survival and reward.

VI. Conclusion

The main motivation of the project was to seek out a minimal biological organism, and potentially demonstrate its computational aptitudes via experiments. The synthetic construction of the minimal biological organism, *C. elegans*, referred to as an agent, clearly exhibited the properties of receiving input from the environment, and processing said input via memory. Through experiments, the agent was able to perform tasks more readily if the agent was

constructed to have a memory, as opposed to being equipped with no working memory. The property of *programmability* was not investigated in this project, but would serve as a future next step.

END REMARKS

The author thanks Professor Jim Crutchfield for facilitating the 2^8AB course sequence, as it was exceedingly refreshing, intellectually stimulating and supremely engaging. The author also thanks Mikhael Semaan, the TA of the course sequence, for his ongoing helpful feedback, advice and suggestions throughout the quarter. Finally, the author thanks Komal Sah and Jacob Hastings for their helpful feedback for improving this project.

REFERENCES

[1] T. Ahamed, A. Costa, and G.J. Stephens. "Capturing the continuous complexity of behavior in *Caenorhabditis elegans*". In: *Nature Physics* (2021).

[2] Google Quantum AI. *Quantum Computer Datasheet*. URL: https://quantumai.google/hardware/datasheet/weber.pdf.

[3] S.J. Cook et al. "Whole-animal connectomes of both *Caenorhabditis elegans* sexes". In: *Nature* (2019).

[4] D-WAVE. *A Practical Approach to Quantum Computing*. URL: https://www.dwavesys.com/learn/d-wave-s-approach/.

[5] IBM. *IBM Quantum Systems*. URL: https://www.ibm.com/quantum/systems.

[6] K. Mori et al. "Probabilistic generative modeling and reinforcement learning extract the intrinsic features of animal behavior". In: *Neural Networks* (2021).

[7] Reference*. *How Many Computers Are There in the World?* URL: https://www.reference.com/world-view/many-computers-world-e2e980daa5e128d0.

[8] A Sharma et al. "A Probabilistic Chemical Programmable Computer". In: *arXiv* (2022). arXiv: 2204.13493 [cs.ET].

[9] Statista. *Forecast number of mobile devices worldwide from 2020 to 2025 (in billions)*. URL: https://www.statista.com/statistics/245501/multiple-mobile-device-ownership-worldwide/.

[10] G.J. Stephens et al. "Dimensionality and Dynamics in the Behavior of *C. elegans*". In: *PLOS Computational Biology* (2008).

[11] Northwestern University. *NetLogo*. URL: https://ccl.northwestern.edu/netlogo/.

[12] W.B. Wood. *The Nematode Caenorhabditis elegans*. Cold Spring Harbor Monograph, 1988.