3a. Goal: What is your primary project goal? What you would like to learn?

I would like to see if internal cybernetic/control properties of a system (with some level of agency, or at least a goal/purpose in the case of engineered systems) are reflected in its intrinsic computational or information-processing capabilities, and examine the relationship of those quantities to the agent's fitness.

I was hoping to analyze and compare/contrast a variety of different systems: various self-modifying software systems (such as Tierra and Avida, also see [3] and [4])-- perhaps even neural networks or standard genetic programming algorithms such as crossover-- all performing the same task, and compare various information-theoretic properties (most ideally h_μ and C_μ) that give some kind of characterization to the process that produced whatever data is collected from these systems, hopefully enabling inferences to be drawn about how these systems work, both from a dynamical perspective (see [0]) and an intrinsic-computational one.

3b. System: Describe how the dynamical system is nonlinear and time-dependent. What's the state space? What's the dynamic? Why is the system behavior interesting?

For each evolving program, the state space are all combinations of instructions defined for the given framework that could be generated, valid or invalid, of any length (probably up to a defined maximum). The exact dynamic will depend on the language/system/algorithm in question, but should reflect that framework's unique path to convergence on an answer to the problem it was tasked with. The system's behavior is interesting because it is completely artificial, and completely reflects its design (structure). Thus we are able to study how very contrived, purposeful design decisions effect function over time in an evolutionary setting. Since many of these systems are spin-offs and close variations of each other (e.g. Cartesian Genetic Programming vs Self-Modifying Cartesian Genetic Programming), it may even be possible to explore the effects of incredibly specific differences.

3c. Dynamical properties: What dynamical properties are you going to investigate?

There are many possibilities: the overall dynamical landscape in the form of attractors, peaks and valleys, discontinuous events (bifurcations etc), as well as speed of convergence to the correct answer for different problems.

3d. Intrinsic computation properties: What information processing properties are you going to investigate?

Some more thought will be needed regarding methods and which specific frameworks/languages/algorithms/etc to use (this will also affect 3c above). The ideal case would be to attempt to construct an ϵ-machine for the process (this may be difficult or impossible), and figure out its entropy rate (h_μ), the statistical complexity (C_μ). I am also curious if synchronization-related quantities have a meaning in this context (considering they seem mostly to apply to periodic processes) and in what way they are related to the program's speed of convergence.

3e. Methods: What methods will you use? Why are they appropriate?

Methods are still up in the air-- I am planning on collecting an assortment of algorithms with some sort of evolutionary component, all trying to solve the same problem (. . . if possible). I would keep track of the program/code generated at each time step, the framework's accuracy at the task and other relevant benchmarks (e.g., time and space performance). Hopefully this would be enough data to fruitfully perform analyses yielding the quantities and qualities mentioned in 3c. and 3d. Although a direct comparison to the "finitary soup" of ϵ-machines from [1] and [2] would not be possible directly, I could compare the results of the analysis from 3d. to those two papers as an additional point of reference.

3f. Hypothesis: What is your current guess as to what you will find?

Genuinely not sure. I am somewhat expecting that algorithms/frameworks that incorporate more features associated with biological life, and that are more open-ended, with fewer arbitrary restrictions (e.g. simulating mutation using specific procedures), to have higher amounts of stored information and information production. I can't really conceptualize how this will relate to actual task performance, but would love to find out.

3g. Steps: List the appropriate steps for your investigation; for example, read literature, write simulator, do mathematical analysis, estimate properties from simulation, write up report, and so on

1. Read literature
    1. Background
        1. review topics of this class (additionally also population dynamics and anything else you recommend)
        2. genetic algorithms (as well as any other framework/algorithm types I choose to study)
        3. cybernetics, population dynamics, early complexity research that would help clarify and sharpen the background model of evolution and how organisms "work" I am using to interpret the results of this project
    2. Understand each algorithm/framework I choose to study in depth-- how it works, its exact output, requirements, etc, as well as its purpose, design, and known mathematical properties
    3. Look for directly related work that has already been done on topics that are as similar as possible
        1. note: so far have not seen this approach applied to self-modifying code/software specifically

2. Design/adjust simulation and analysis in greater detail after thorough literature review
3. Set up and run simulation and benchmarking pipeline
4. c. Analyze dynamical properties and d. analyze intrinsic computation properties
5. Write report

3h. Time: Estimate how long each step will take. Can you complete the project within one month?

If I was able to work on the project 8 hours a day for the whole month, chances would be good that it would be finished in that time frame. As it currently stands, it seems unlikely. 1, if some topics are eliminated and with skimming, would probably take 1-2 weeks. 2 and 3 combined would probably take 2-3 weeks, each subsequent step also around 2 weeks.

I suspect that a careful choice of genetic algorithms and design simulations/experiments could greatly reduce the amount of time that 3, 4, and 5 would take. Being able to build off of similar prior projects/literature would also help (but so far I have not looked as much as I probably should have). I would greatly appreciate suggestions on how to reduce the scope while still examining the key issue at hand (how exactly self-modification/self-regulation affects the dynamical and intrinsic-computing properties of a system).

[0] Jaeger J, Monk N. Bioattractors: dynamical systems theory and the evolution of regulatory processes. J Physiol. 2014;592(11):2267-2281. doi:10.1113/jphysiol.2014.272385

[1] Olof Görnerup and J. P. Crutchfield, "Primordial Evolution in the Finitary Process Soup", Electronic Journal of Theoretical Physics 4:16I (2007) 297-311. Reprinted in Physics of Emergence and Organization, Eds. I. Licata and A. Sakaji, World Scientific (New Jersey, 2008) 297-311.

[2] Crutchfield James P and Görnerup Olof 2006 Objects that make objects: the population dynamics of structural complexityJ. R. Soc. Interface.3345–349

[3] Sadia Noreen, Shafaq Murtaza, M. Zubair Shafiq, and Muddassar Farooq. 2009. Evolvable malware. In Proceedings of the 11th Annual conference on Genetic and evolutionary computation (GECCO '09). Association for Computing Machinery, New York, NY, USA, 1569–1576. DOI:https://doi.org/10.1145/1569901.1570111

[4] Williams, Lance. (2014). Self-Replicating Distributed Virtual Machines. 10.7551/978-0-262-32621-6-ch114.

[5] Lenski, R., Ofria, C., Pennock, R. et al. The evolutionary origin of complex features. Nature 423, 139–144 (2003). https://doi.org/10.1038/nature01568