Goal for each pre:
5 minutes information
2 minutes “treasure”
3 minutes Discussion
---
10 minutes for logistics, feedback

Week 4

CSSE290
Artificial Life

01

Presenter

Will + Steven

Good information:

- "Any system that meets the three basic requirements of Heredity, Variation and Selection will result in evolution." (Harvey, 1)
- "The most creative and challenging parts of programming a GA are usually the problem-specific aspects. What is the problem space, how can one best design the genotypic expression of potential solutions or phenotypes, and the genotype-phenotype mapping, how should one design an appropriate fitness function to achieve one's goals?" (Harvey, 2)

Will + Steven

Microbial Genetic Algorithm:

- "combines the random undirected parent-picking with the directed selection of who is to be culled. Pick two individuals at random to be parents, and generate a new offspring from them; then use the same two individuals for the tournament to select who is culled — in other words the weaker parent is replaced by the offspring. It turns out that this is easy to implement, is effective. This is the underlying intuition behind the Microbial GA, so called because we can interpret what is happening here in a different way — **Evolution without Death!** " (Harvey, 5)

Will + Steven

Treasure:

- Genetic algorithms can be used in many situations including physical wings!

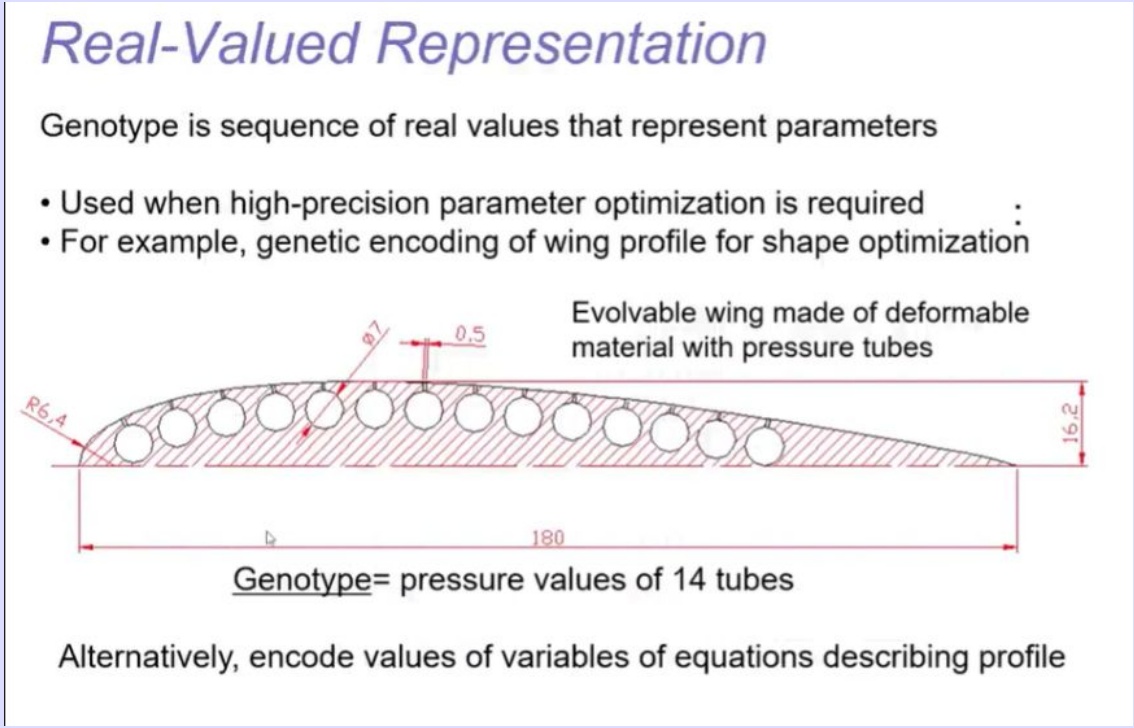


Image from Dr. Yoder's Bio-AI videos

02

Presenter

Self-Replication

by Dominic Reilly

Langton's Ant

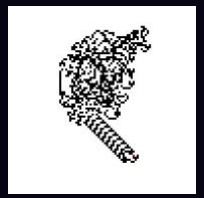
2D Turing Machine / Cellular Automata

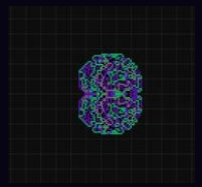
- Cells can be white or black
- Ant moves forward every timestep, flipping the color of the square he is on
- Turns 90 degrees clockwise on a white square
- Turn 90 degrees counter-clockwise on a black square

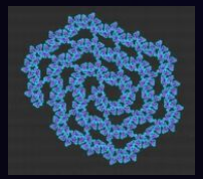
Behaviors

- Start off with simple symmetrical patterns
- Spirals into chaos for ~10,000 steps
- Repeating pattern emerges
- 104 step highway

Can be extended to more than 2 colors



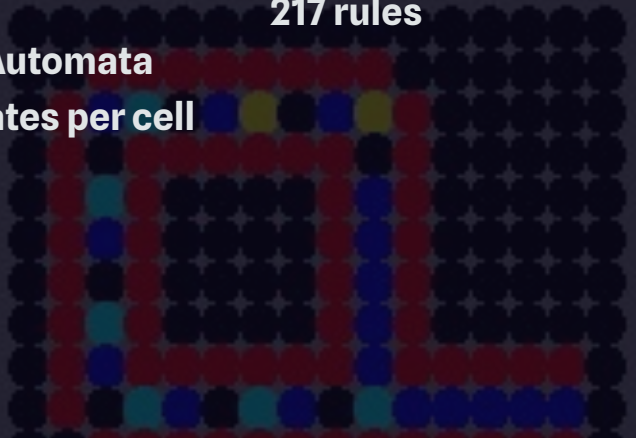




Langton's Loops

Cellular Automata
with 8 states per cell

217 rules



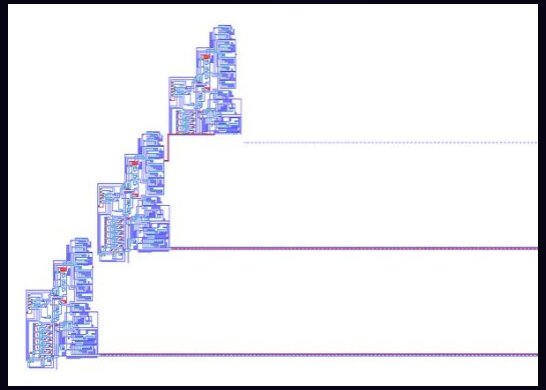
Loops that create
copies of themselves

Expands outward
forever

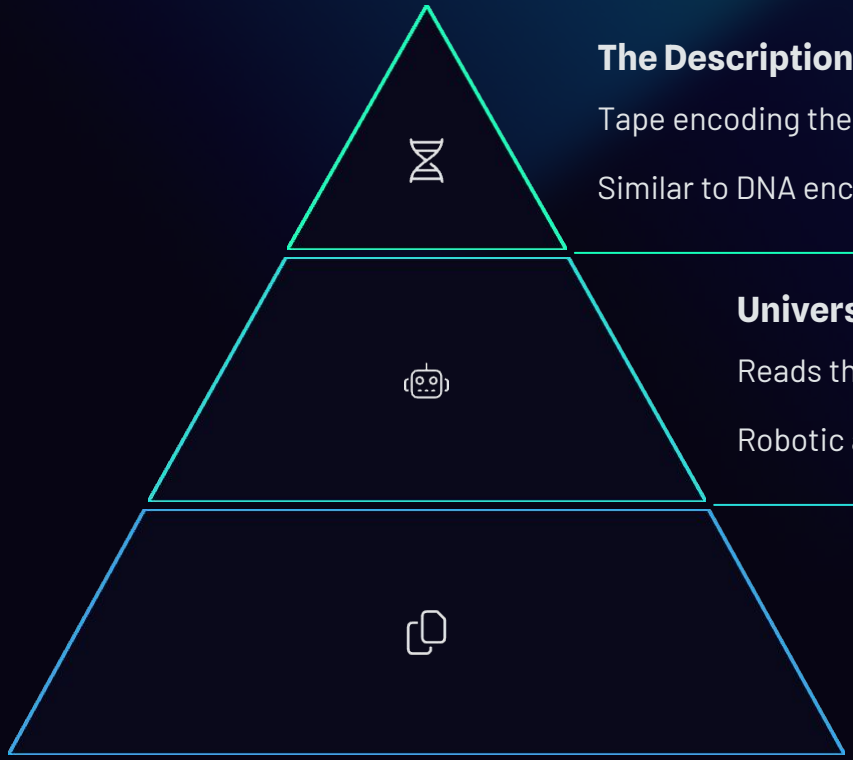
von Neumann's Universal Constructor

What is it?

- 2D cellular automata with 32 possible states per cell
- Also a Turing Machine
- Capable of creating anything that you can encode within its description



3 Parts



The Description

Tape encoding the information needed to create a new machine

Similar to DNA encoding an organism

Universal Constructor

Reads the blueprint and create the machine that is encoded within it

Robotic arm building structures

Universal Copier

Copies the blueprint into the new machine

Continue the cycle of self-replication

Evolution

Mutation

Introduce mutation in blueprint copying

Selection

Fit machines survive

Replication

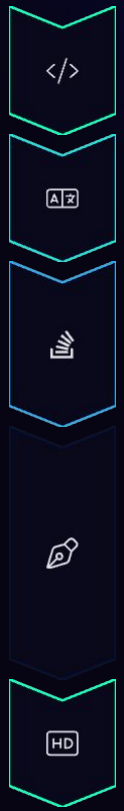
Fit machines replicate

Evolution

Fitness improves over time

Fitness function is the ability to self-replicate

PushGP



Stack-based programming system

Push utilized for genetic programming (GP)

Multiple implementations

Has been implemented in many programming languages (C, lisp, Python)

Type-specific stacks

Every data type has its own stack

- Integers, strings, executable code

Execution process

Genomes represent starting executable code to be pushed to the exec stack

Pop the top executable and run it

- This likely will push data to the other stacks such as int stack
- Future executables can pop this data from the other stacks

Evolutionary evaluation

Genomes are evaluated according to their error and evolutionary algorithms are applied

PyshGP Results: And Gate

Best Seen Individual

Genome:

pvector([Input(input_index=0), Literal(value=0, push_type=<pyshgp.push.types.PushIntType object at 0x0000018FFFD06E70>), Input(input_index=1), InstructionMeta(name='int_yank', code_blocks=0)])

Program:

(input_0 0 input_1 int_yank)

Error vector:

[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

Total error:

0.0

PyshGP Results: Rectangle Area Difference

Solution found.

Best Seen Individual

Genome:

pvector([InstructionMeta(name='exec_dup', code_blocks=1), InstructionMeta(name='rectangle_area_diff', code_blocks=0), Input(input_index=1), Input(input_index=0)])

Program:

(exec_dup (rectangle_area_diff input_1 input_0))

Error vector:

[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

Total error:

0.0

PyshGP Results: Pythagorean Theorem

Best Seen Individual

Program:

(2.0 square_root_func)

Error vector:

[0.18965771 0.4903543 0.56133903 0.01657838 0.24648955 0.1408354
0.31023448 0.39736874 0.43016764 0.53911683 0.26565455 0.19596679
0.76540418 0.14442105 0.16692497 0.12313262 0.01918056 0.07604058
0.02965324 0.63993649]

Total error:

5.748457069491287

03

Presenter

Person Name

Talking Points

Wrapping Up

Connection to prior weeks?

Provide Peer Evaluation (including Self)

Portfolio Reflection Entry