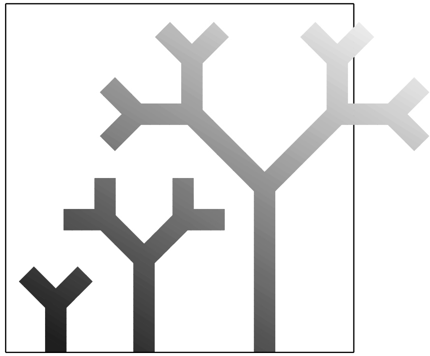
Morphological development and evolution



Companion slides for the book *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies* by Dario Floreano and Claudio Mattiussi, MIT Press

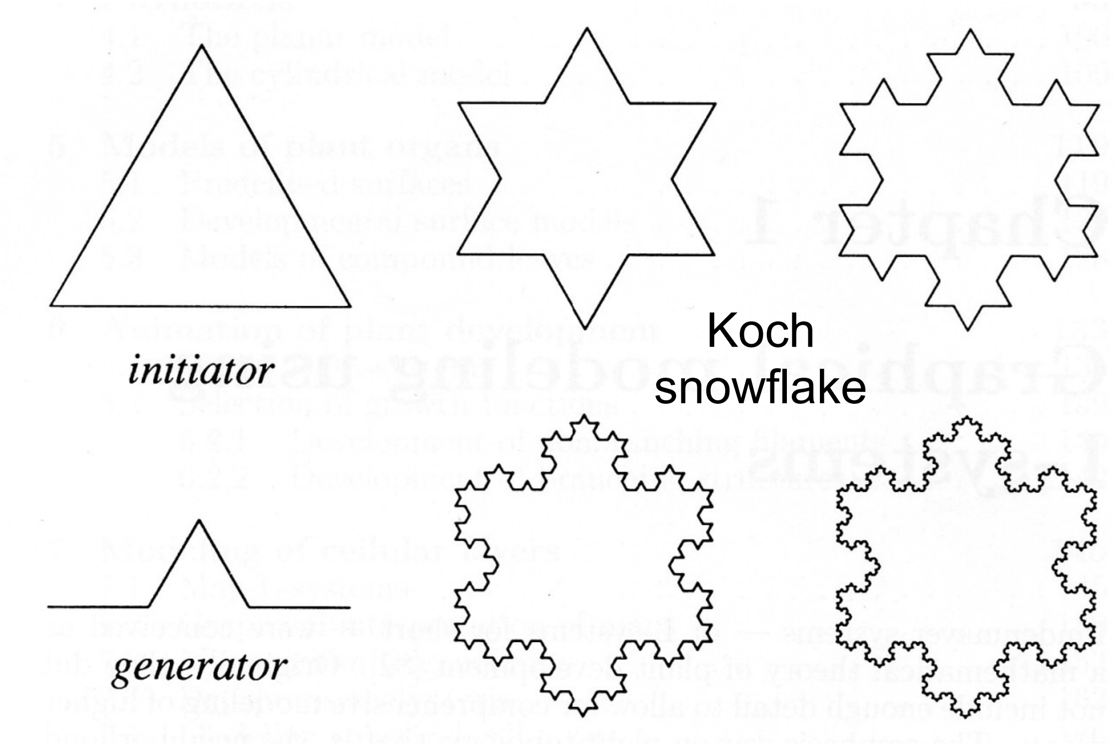
What you will learn in this class

- Represent complex structures as a growth process
- How to encode plant-like structures
- Encoding and evolution of neural architectures
- Encoding and evolution of robotic bodies and brains
- Composition Pattern Producing Networks
- Morphological computation: how bodies simplify control
- Co-evolved bodies make learning faster and better

Growth by Rewriting

Rewriting System: recursively replace a sub-component with another sub-component

Fractals: *Replace edges of a polygon with open polygons and rescale at each iteration [von Koch, 1905]*



Several types of rewriting systems have been developed. For example:

L-systems (plants)

Cellular automata (anything)

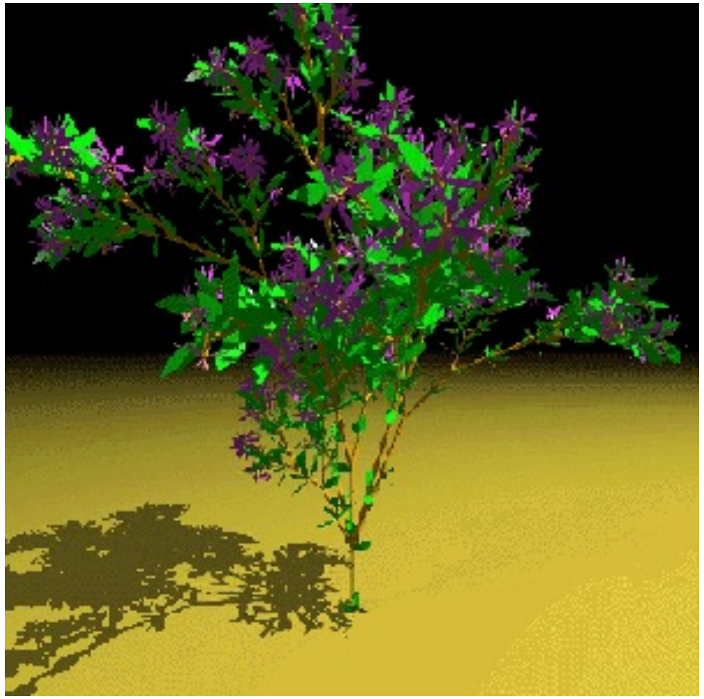
Language systems (language)

Matrix rewriting (neural networks)

L-systems [Lindenmayer, 1968]

Lindenmayer systems, or L-systems for short, are mathematical models to describe biological morphologies through a growth process. They were originally applied to model growth of plants.

Aristid Lindenmayer



http://local.wasp.uwa.edu.au/~pbourke

Artificially generated tree

L-system: Definition

L-systems are rewriting systems that operate on symbol strings.

An L-system is composed of:

1. A set of symbols s forming an *alphabet* A
2. An *axiom* ω (initial string of symbols) $s_{k,}\ s_{z,}\ s_{v,\dots}$
3. A set $\pi = \{p_i\}$ of *production rules* $p_i : s_k \rightarrow s_z$.

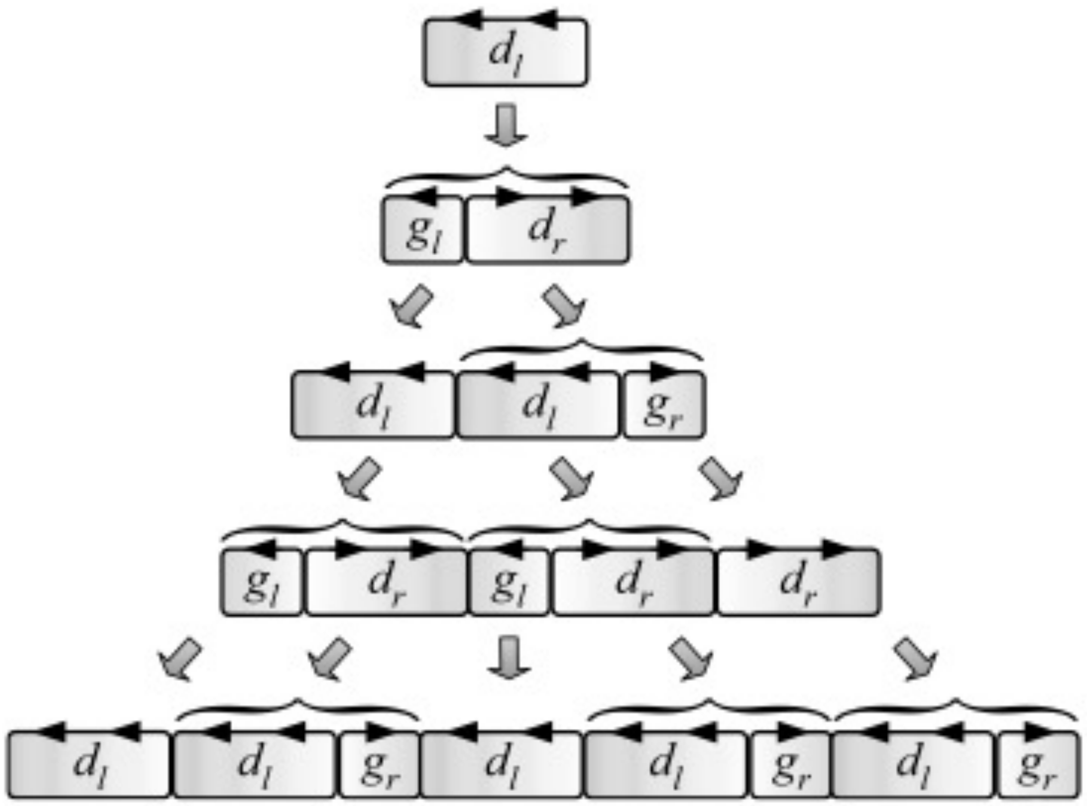
The following assumptions hold:

1.Production rules are applied in parallel and replace recursively all symbols in the string.

2.If no production rule is specified for a symbol s, then we assume the identity production rule $p_o : s_k \rightarrow s_k$

L-system: 1D Example

Development of a multicellular filament of blue-green bacteria *Anabaena catenula* [Lindenmayer 1968]



Cells can be in a "growing" state g or in a "dividing" state d with left or right polarity

$$A = \{g_r, g_l, d_r, d_l\}$$

$$\omega = d_l$$

$$p_1 = d_r \rightarrow d_l\, g_r$$

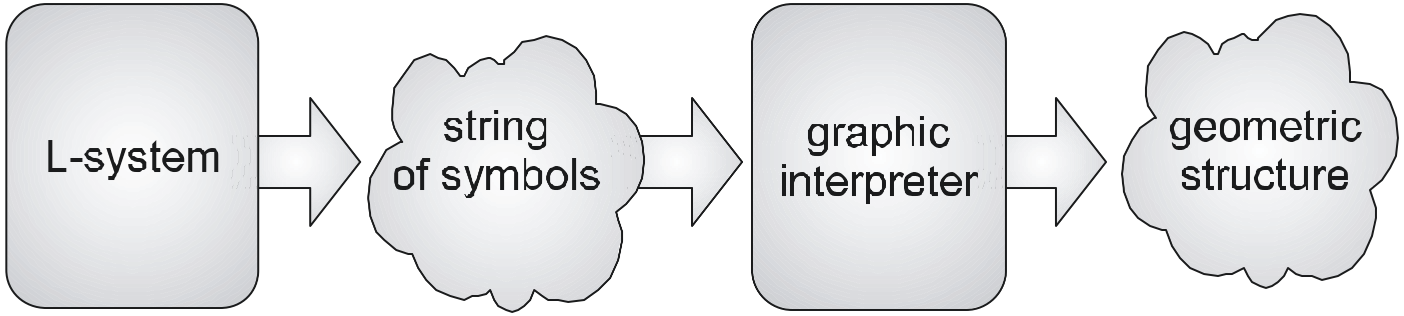
$$p_2 = d_l \rightarrow g_l\, d_r$$

$$p_3 = g_r \rightarrow d_r$$

$$p_4 = g_l \rightarrow d_l$$

Graphics Interpretation

- Using symbols that represent directly geometric entities such as 1D or 2D cells becomes rapidly impractical.
- We can increase the graphic potential of L-systems by following the phase of production of strings of symbols with a phase of graphic interpretation of the strings



Turtle Graphics Interpretation

In 2D, the turtle (printer) state is defined by the triplet x, y, α where the Cartesian coordinates (x, y) represent the turtle's position and the angle α, also known as heading, represents the facing direction.

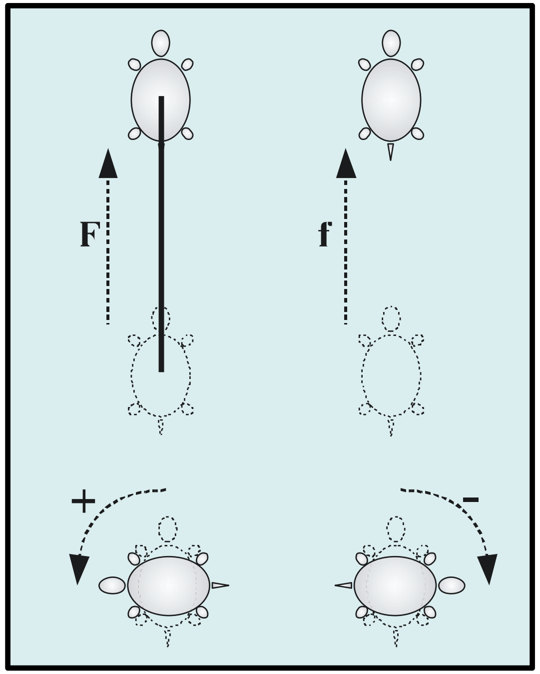
Given the step size d and the angle increment δ, the turtle can respond to the following commands:

F : move forward by a step while drawing a line.

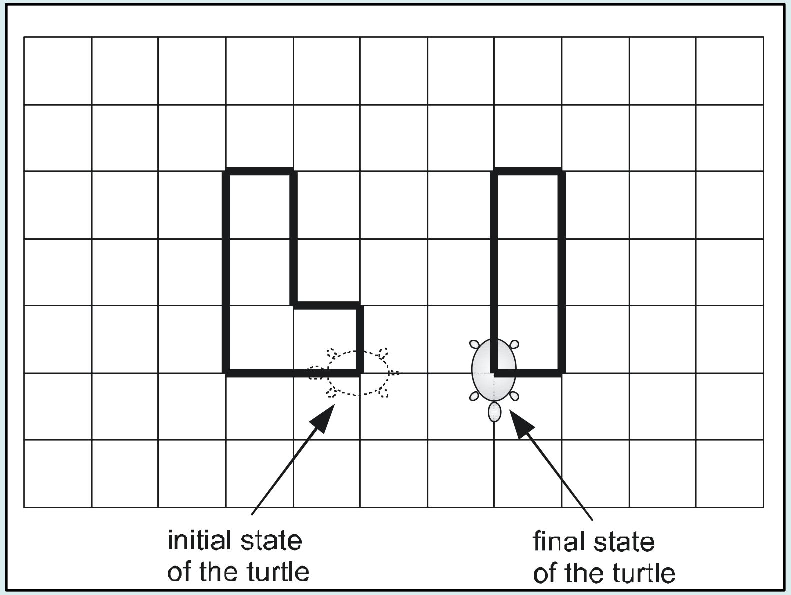
f : move forward by a step without drawing a line.

+ : turn left (counterclockwise) by angle δ.

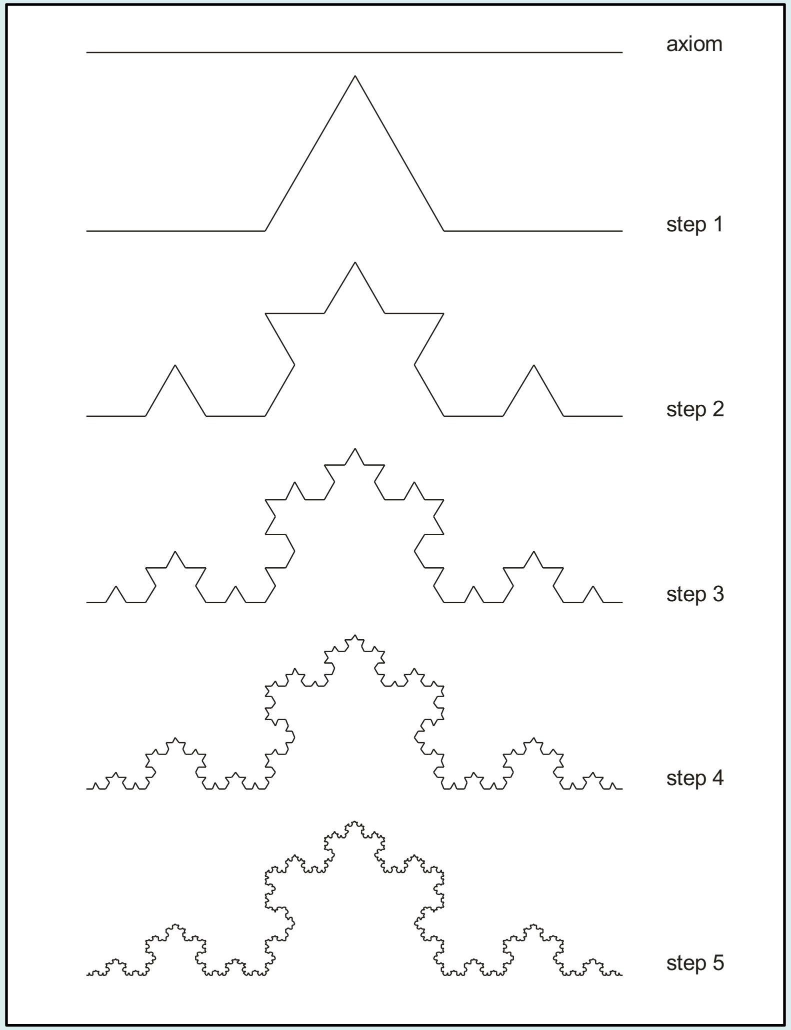
– : turn right (clockwise) by angle δ.



Examples



$$\delta = 90^\circ \quad , \ A=\{ F, f, +, - \}$$
$$\omega = FF - FFF - F - FF + F - F + f f F + FFF + F + FFF$$



$$\delta = 60^\circ \quad , \ A=\{ F, f, +, - \}, \ \omega = F$$
$$p = F \rightarrow F{+}F{-}{-}F{+}F$$

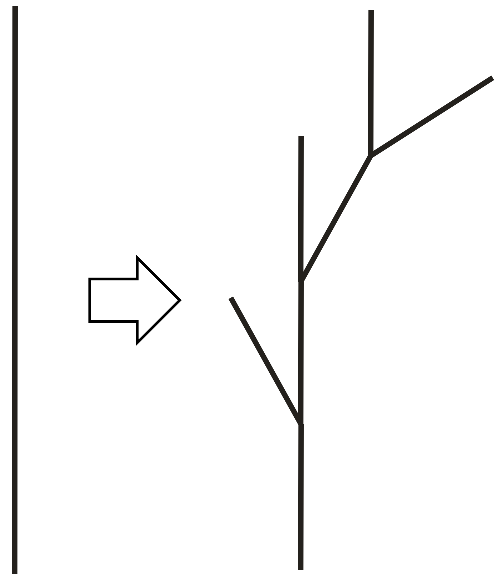
Bracketed L-systems

In drawing branching structures using the turtle interpreter it is necessary to reposition the turtle at the base of a branch after the drawing of the branch itself

- Two new symbols:

[Save current state of the turtle (position, orientation, color, thickness, etc.).

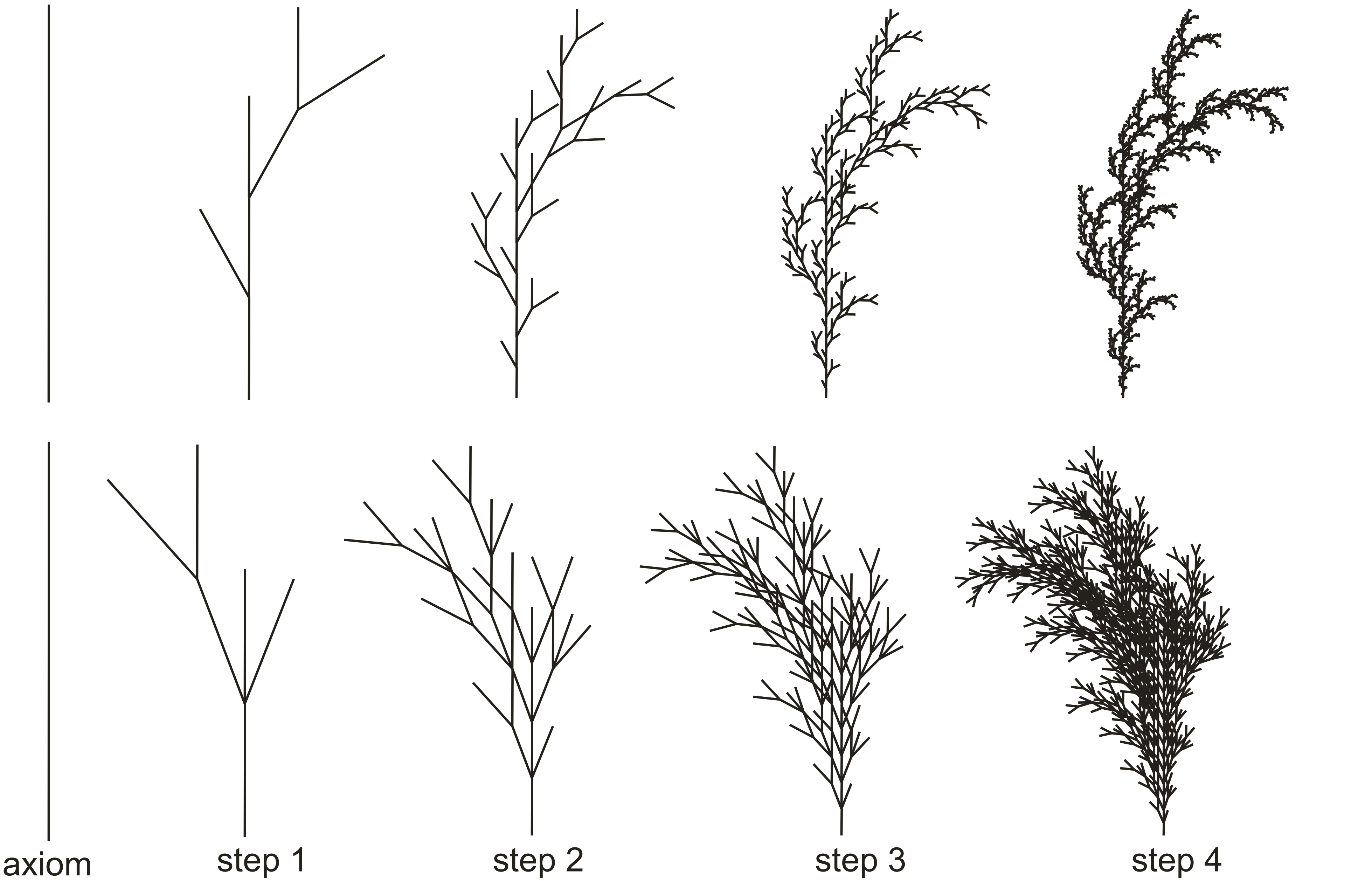
] Restore the state of the turtle using the last saved state (no line is drawn).

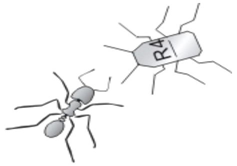


δ $\delta = 29^\circ$, $A = \{ F, +, -, [,] \}$

$\omega = F$

$p = F \rightarrow F[+F]F[-F[+F][-F]]F$


axiom
step 1
step 2
step 3
step 4



Stochastic L-systems

- In nature individuals of the same species are not identical.
- Specimen variability can be modeled by associating probabilities to production rules
- The sum of all probabilities over the same symbol must be 1

$\delta \quad \delta = 29^\circ \;,\; A = \{ F, +, -, [,] \}$

$\omega = F$

$$p_1 = F \xrightarrow{1/3} F[+F]F[-F]F$$

$$p_2 = \mathrm{F} \xrightarrow{1/3} F[-F]F[+F]F$$

$$p_3 = \mathrm{F} \xrightarrow{1/3} F[-FF-F]F$$

Application to computer graphics

http://gug.sunsite.dk/

http://www.ii.uib.no/~knute

http://www.uweb.ucsb.edu/~svetlin

How to identify rewriting systems?

- By hand
  - When the rewriting rules are explicitly given (e.g., fractal curve)
  - When the rewriting rules can be easily deduced from the description of the developmental process (e.g., development of bacteria filaments and moss leaves)
  - When the resulting morphologies have only an aesthetic function

- With heuristic search methods (e.g., evolutionary computation)
  - When the details of the resulting morphology have a functional role, such as a neural network, a gene regulatory network, an electronic circuit)

Neural architecture by matrix rewriting

A *rewriting system* [Kitano, 1990] that encodes a grammar to represent network topologies

Genome encodes the rewriting (grammar) rules, such as: *ABCD adaa cbba baac abad 0001 1000 0010 0100*

$$\mathcal{E} \rightarrow \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

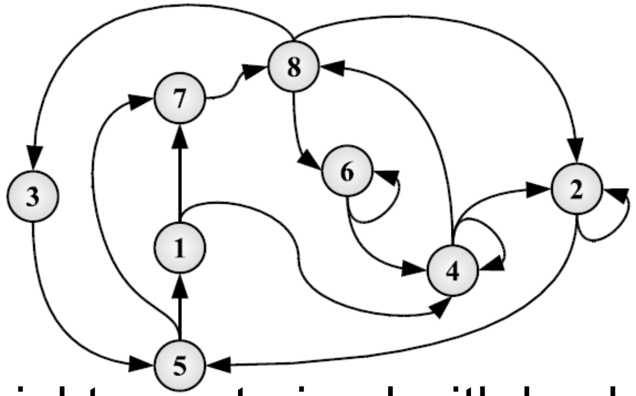
$$A \rightarrow \begin{bmatrix} a & d \\ a & a \end{bmatrix} \quad B \rightarrow \begin{bmatrix} c & b \\ b & a \end{bmatrix} \quad C \rightarrow \begin{bmatrix} b & a \\ a & c \end{bmatrix} \quad D \rightarrow \begin{bmatrix} a & b \\ a & d \end{bmatrix}$$

$$a \rightarrow \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \quad b \rightarrow \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \quad c \rightarrow \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \quad d \rightarrow \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$\mathcal{E} \Rightarrow \begin{bmatrix} A & B \\ C & D \end{bmatrix} \Rightarrow \begin{bmatrix} a & d & c & b \\ a & a & b & a \\ b & a & a & b \\ a & c & a & d \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\begin{array}{c c} & \begin{array}{cccccccc} \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} & \mathbf{6} & \mathbf{7} & \mathbf{8} \end{array} \\ \begin{array}{c} \mathbf{1} \\ \mathbf{2} \\ \mathbf{3} \\ \mathbf{4} \\ \mathbf{5} \\ \mathbf{6} \\ \mathbf{7} \\ \mathbf{8} \end{array} & \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

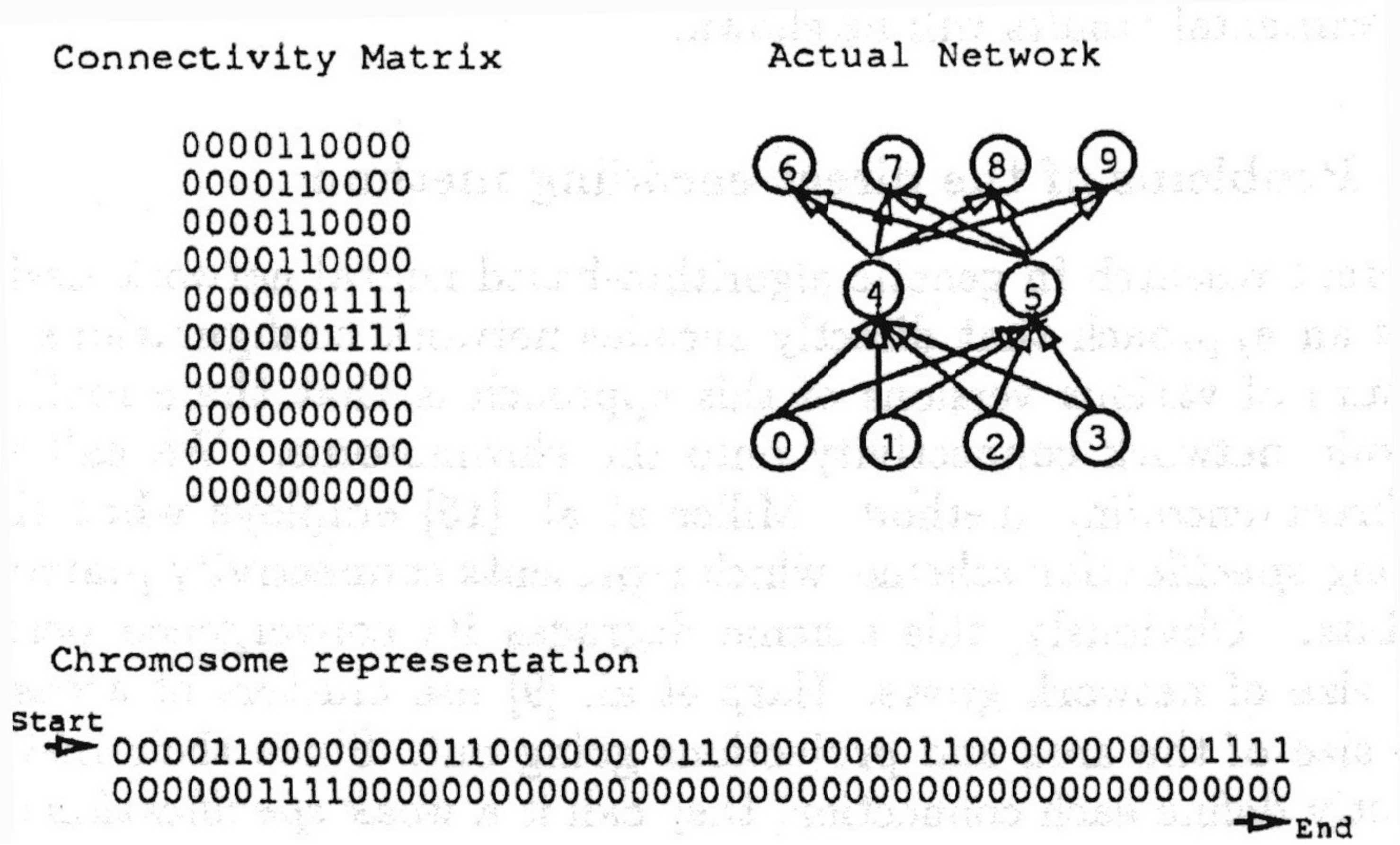
$$\Downarrow$$



Only the presence/absence of connections is evolved. Weights are trained with backpropagation.

Direct Encoding of Network Topology

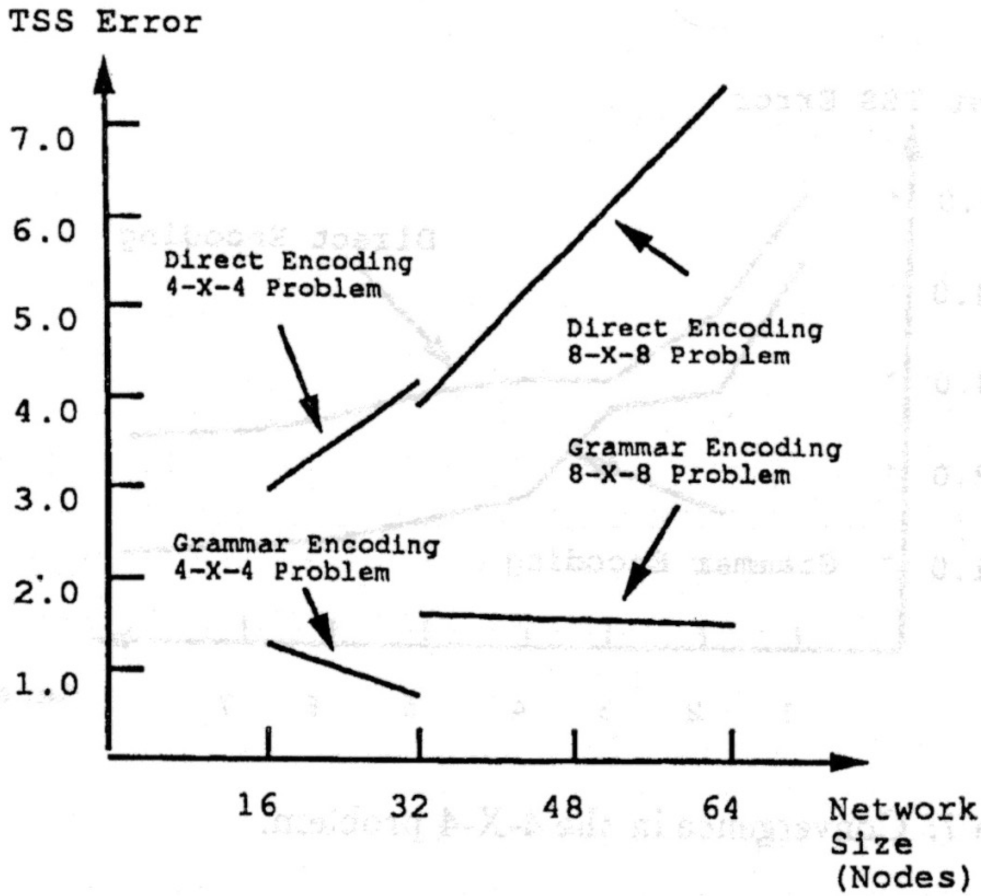
Length of genetic code is proportional to number of neurons in the network



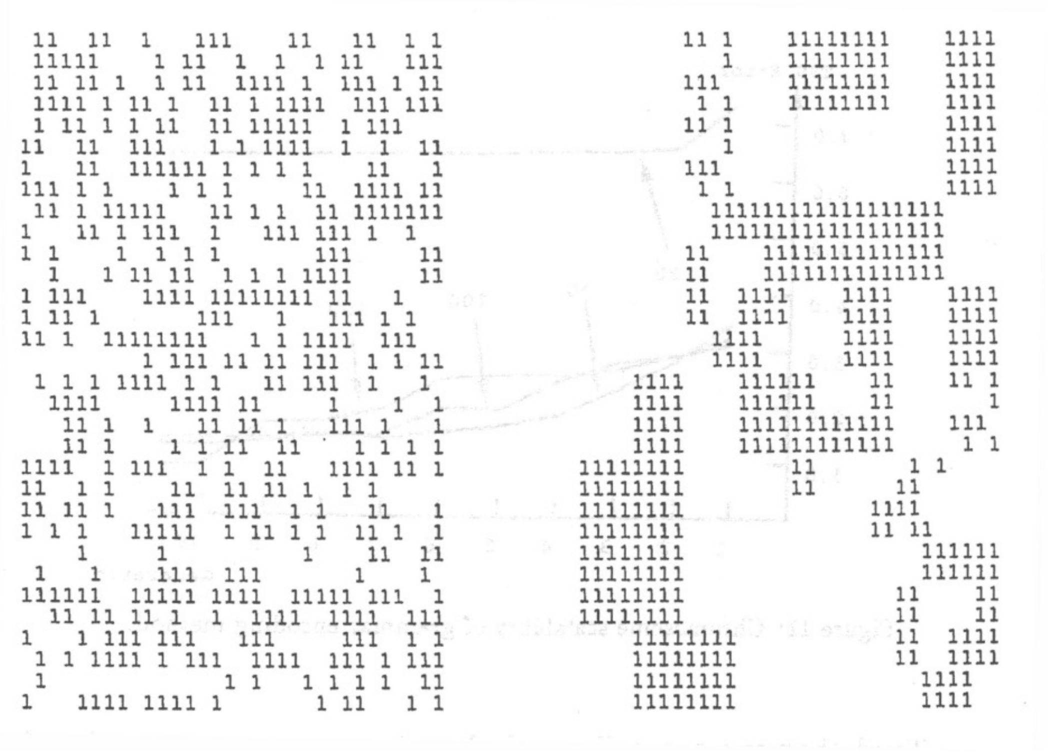
Evolving autoencoder architectures

- Autoencoder performance is worse with direct encoding and worsens with network size
- Topologies evolved with grammar encoding are more regular (good for spatial information processing, such as convolutional neural networks).

Performance comparison



Architecture comparison

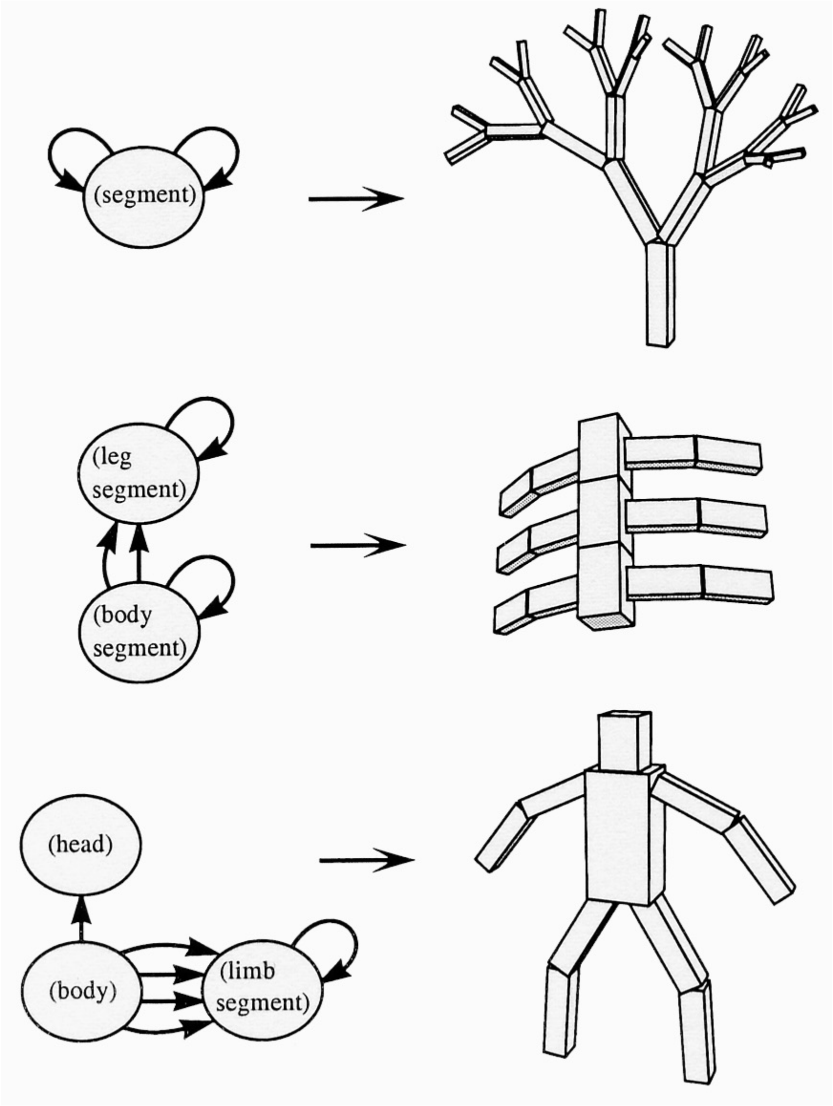


Direct encoding

Grammar encoding

Grammar encoding of robotic bodies and brains

[Sims, 1994]



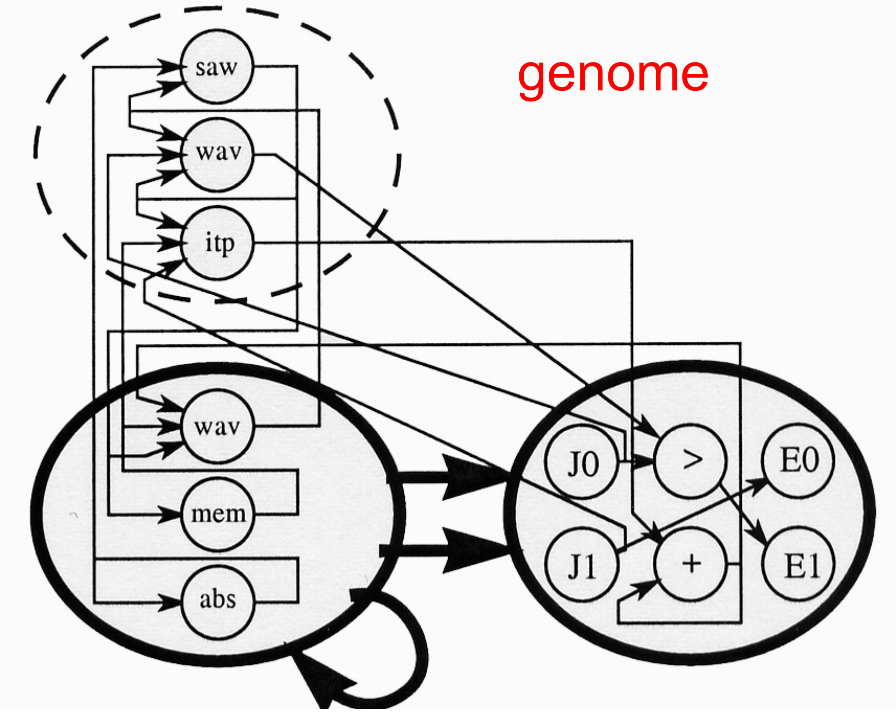
genotype phenotype

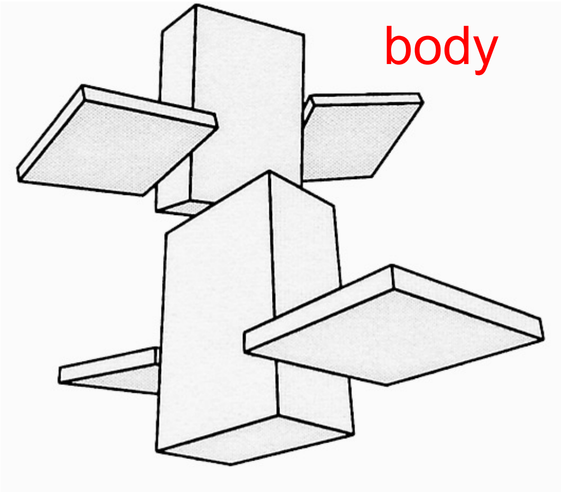
Body components:

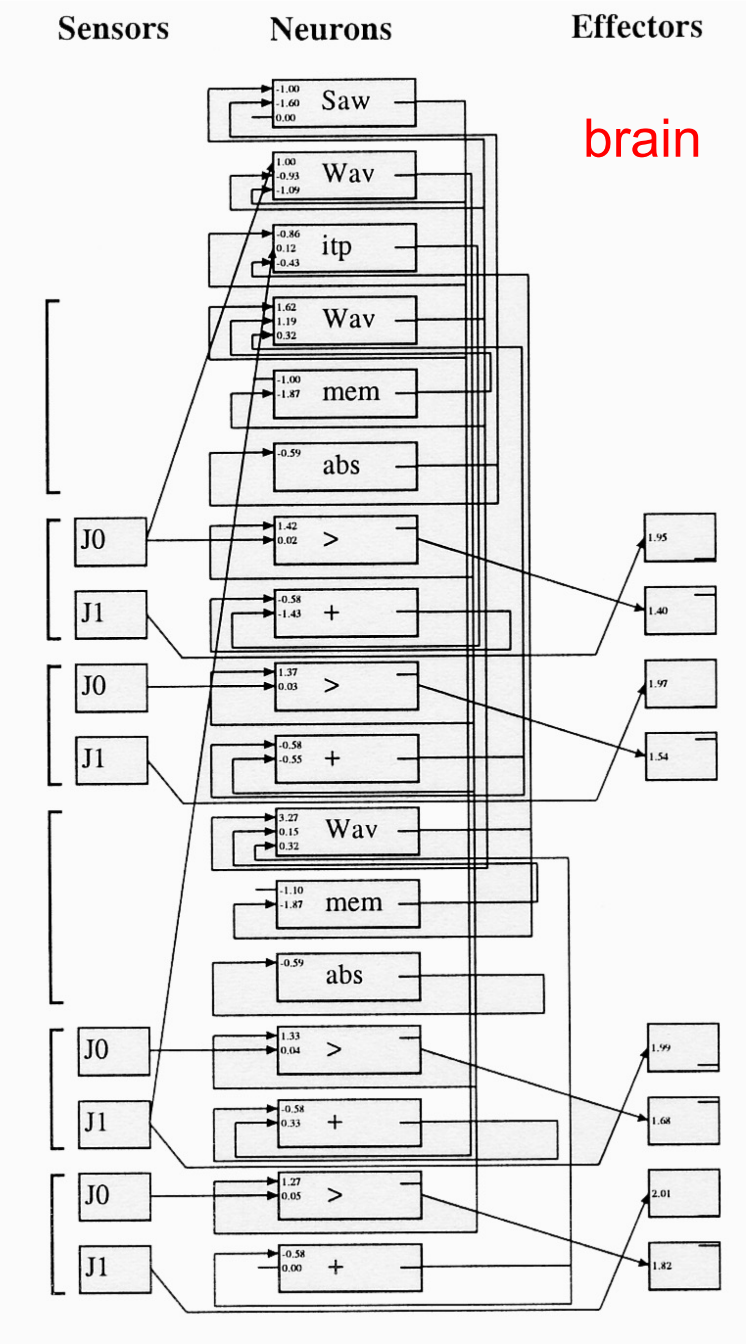
- dimension
- joint type (rigid, twist, revolute, ...)
- recursive-limit
- connection (position, orientation, scale, reflection)
- terminal
- neural circuit

Neural circuit components:

- sensors: rotation, contact, light
- neurons: sum, memory, oscillator, max, etc.
- effectors: push, pull


genome
saw
wav
itp
wav
mem
abs
J0
>
E0
J1
+
E1


body


Sensors
Neurons
Effectors
brain
Saw
Wav
itp
Wav
mem
abs
J0
>
J1
+
J0
>
J1
+
Wav
mem
abs
J0
>
J1
+
J0
>
J1
+

[Sims, 1994]

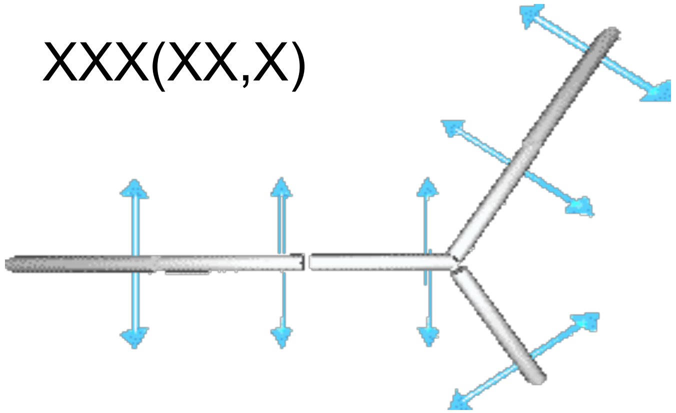
Co-evolved robotic bodies and brains

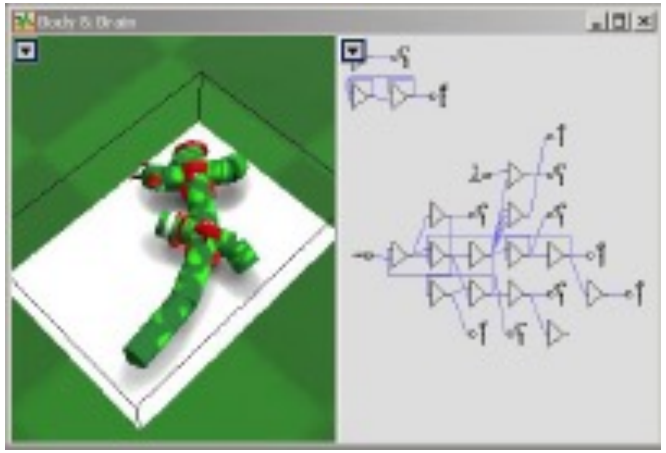
Sims, *1994*

Framstick [Komosinski & Ulatowski, 1999]

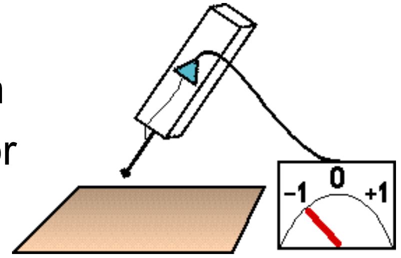
Body parts are joined sticks. Sticks can host sensors and neurons. Joints are actuated by muscles.

XXX(XX,X)

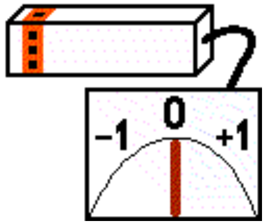


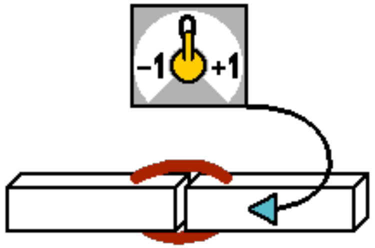


touch sensor

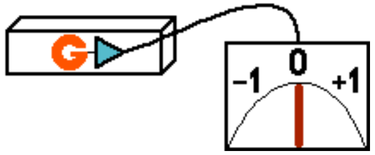


food sensor





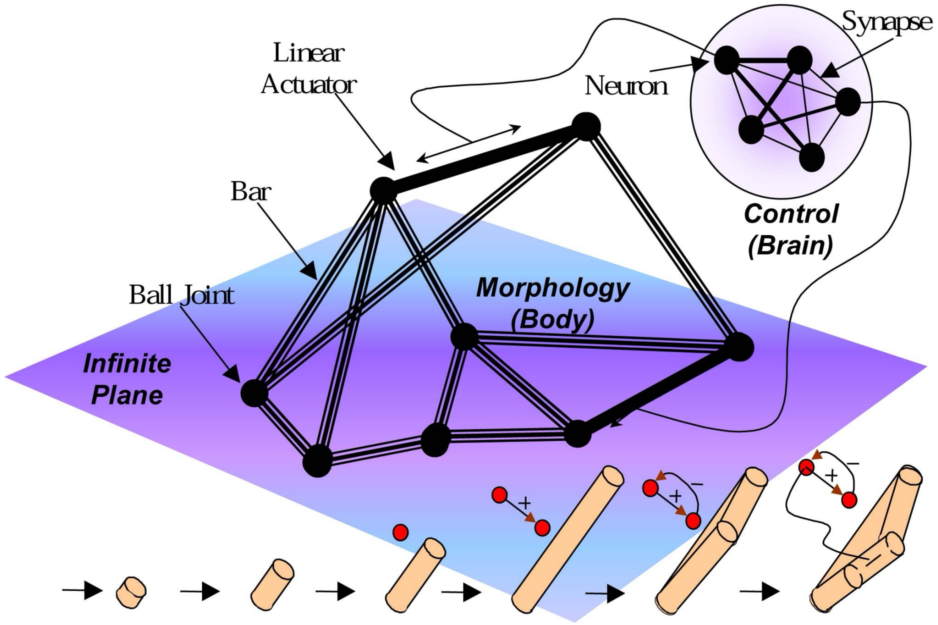
muscle

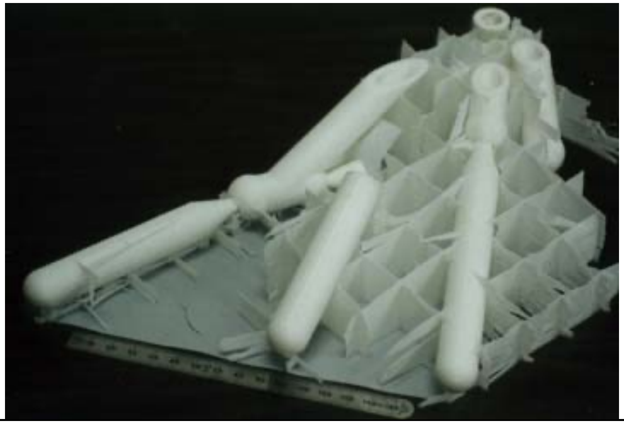


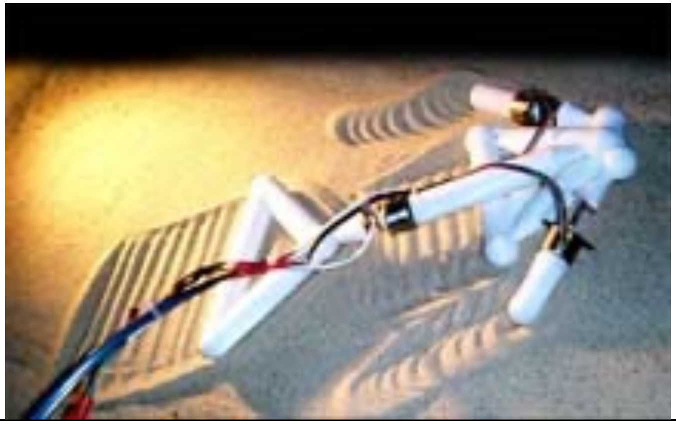
gyroscope

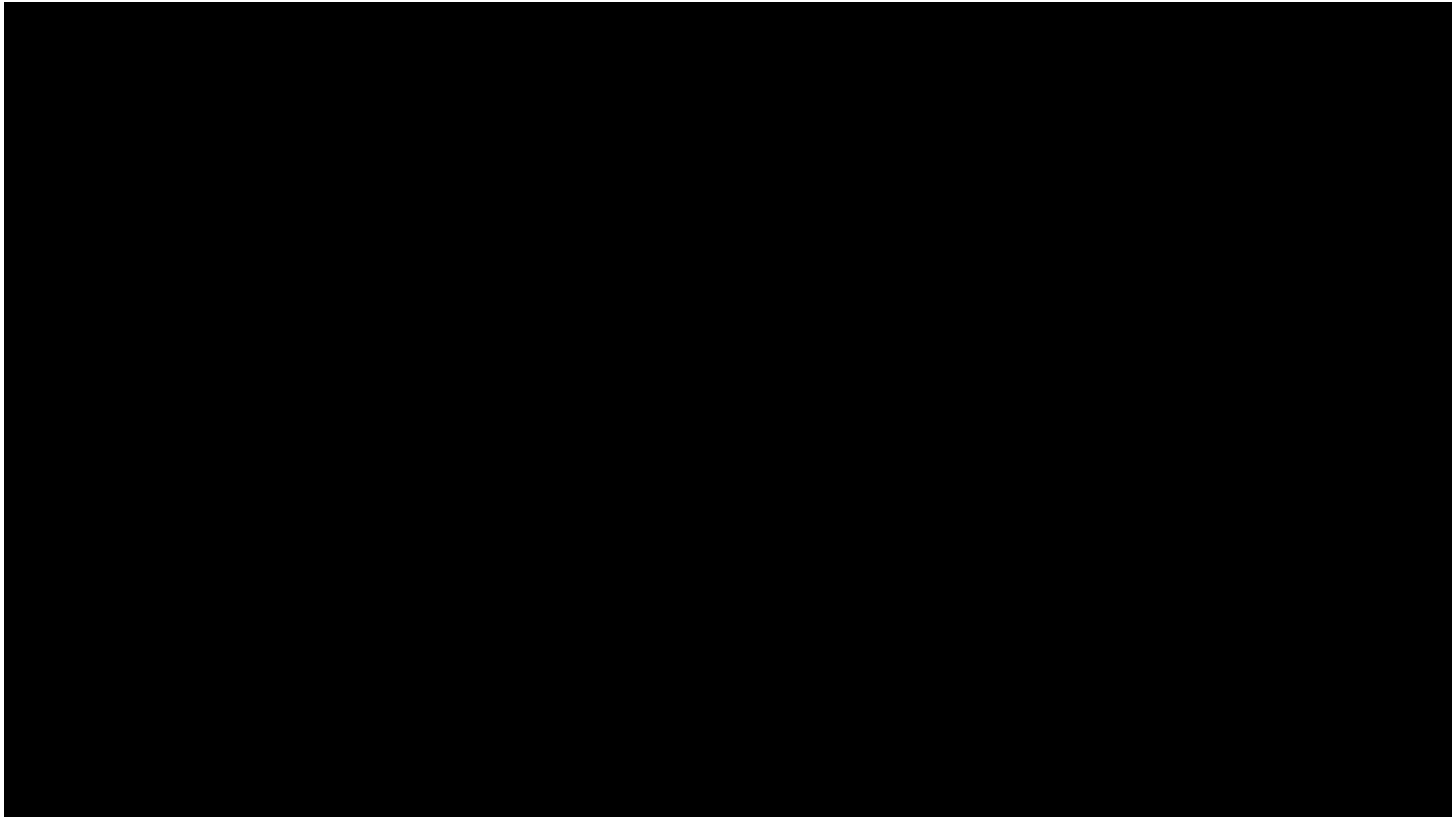
www.frams.alife.pl

The Golem project (Lipson & Pollack, 2000)





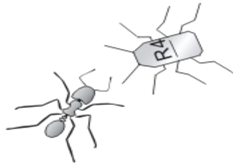




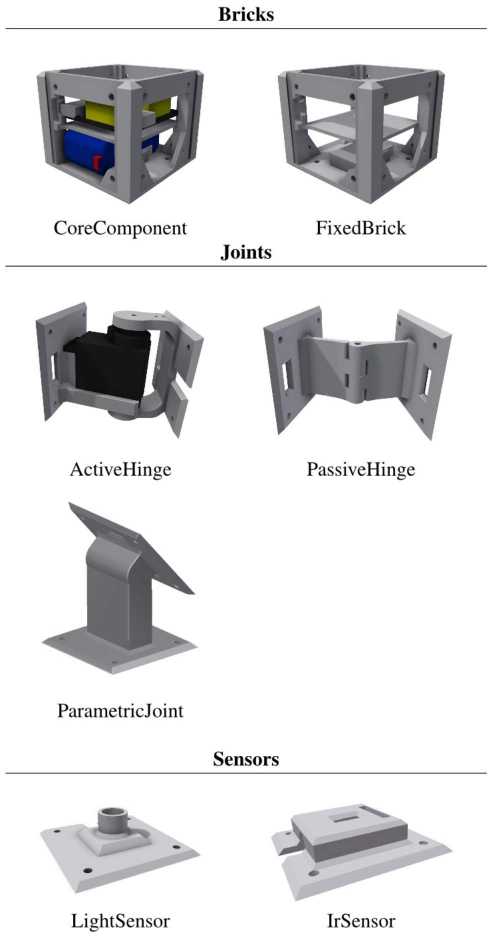
http://www.demo.cs.brandeis.edu/golem/

Robogen

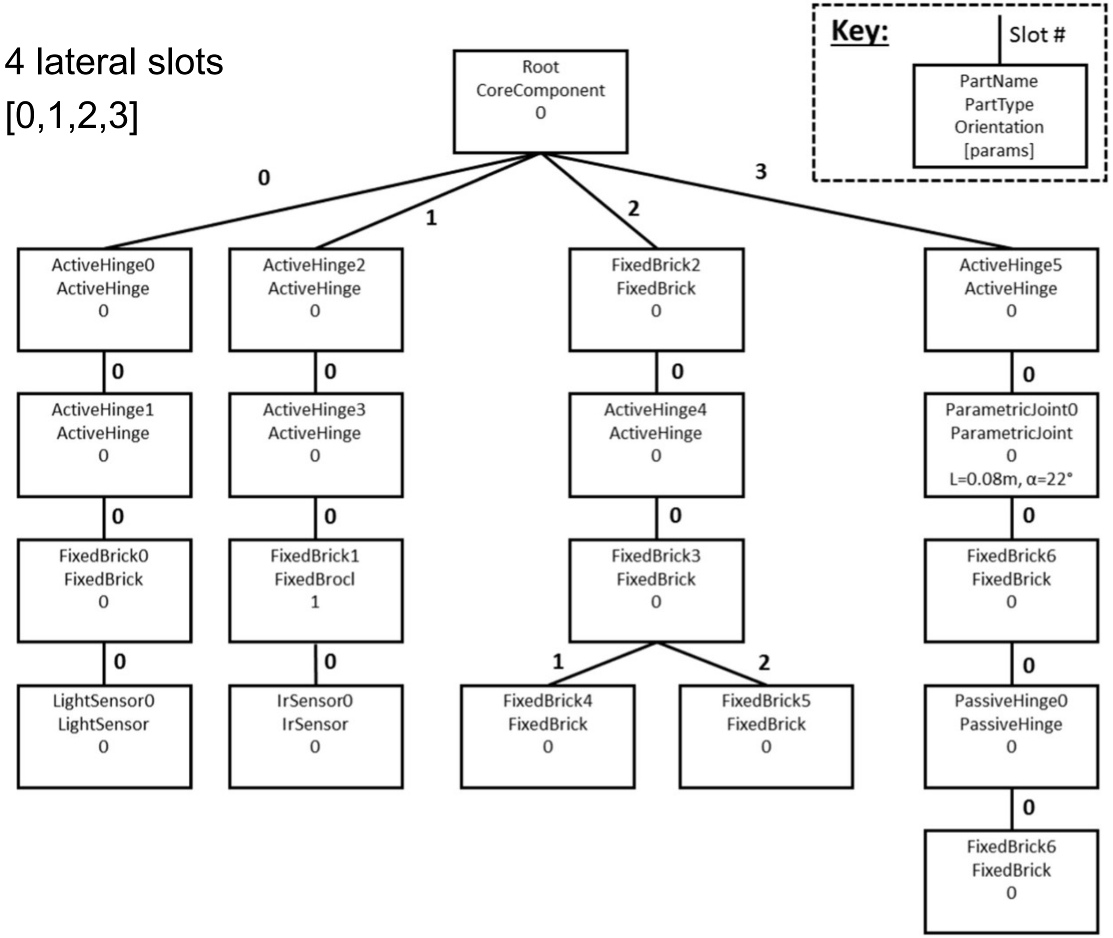
Auerbach J. E., Concordel A., Kornatowski P. M., Floreano D. (2019) Inquiry-Based Learning with RoboGen: An Open-Source Software and Hardware Platform for Robotics and Artificial Intelligence. *IEEE Transactions on Learning Technologies* (12, 3), 356-369.

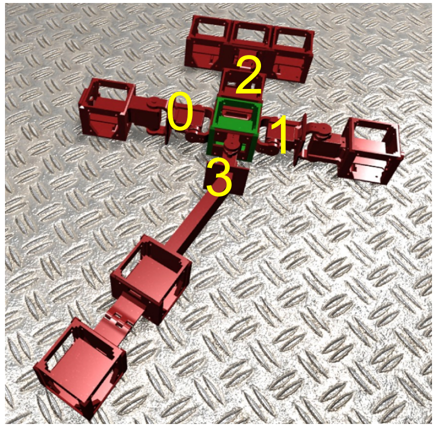


Robogen: Morphology Encoding and Mutations



4 lateral slots
[0,1,2,3]



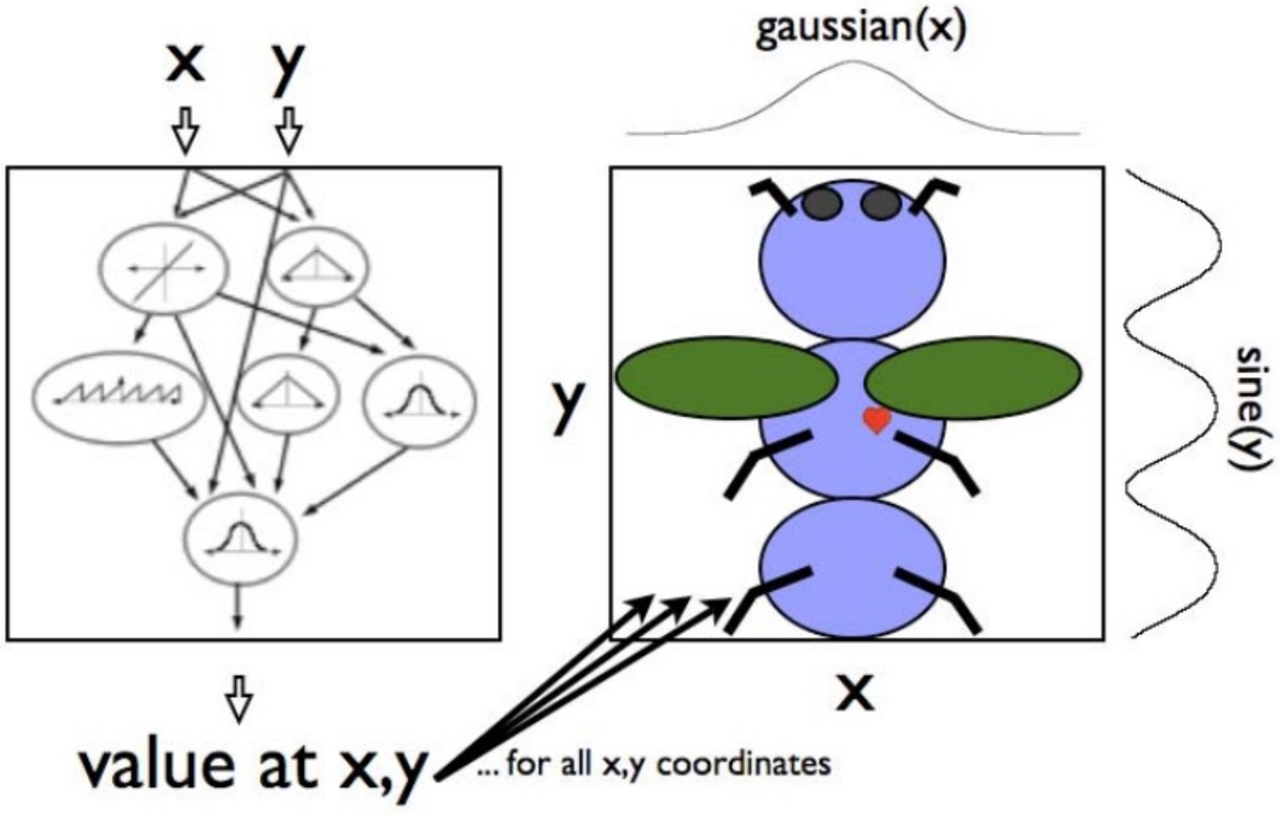


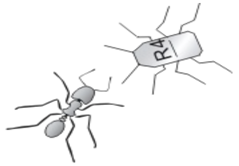
Mutation Operator	Description
NodeInsert	Insert a random node at a random location in the body representation tree.
NodeRemove	Remove a random node from the body tree representation.
SubtreeDuplicate	Duplicate a randomly chosen subtree and insert it at a random location on the body tree.
SubtreeSwap	Swap two randomly chosen subtrees of the body tree representation.
SubtreeRemove	Remove a randomly chosen subtree from the body tree representation. Unlike *NodeRemove* which attempts to remove a node and propagate its children upwards, *SubtreeRemove* removes a node and all of its descendants.
MutateParam	Mutate a randomly chosen parameter of a randomly chosen node. For the purpose of this operator a node's orientation relative to its parent is also consider to be a parameter.

The probability of applying each operator is user-configurable.

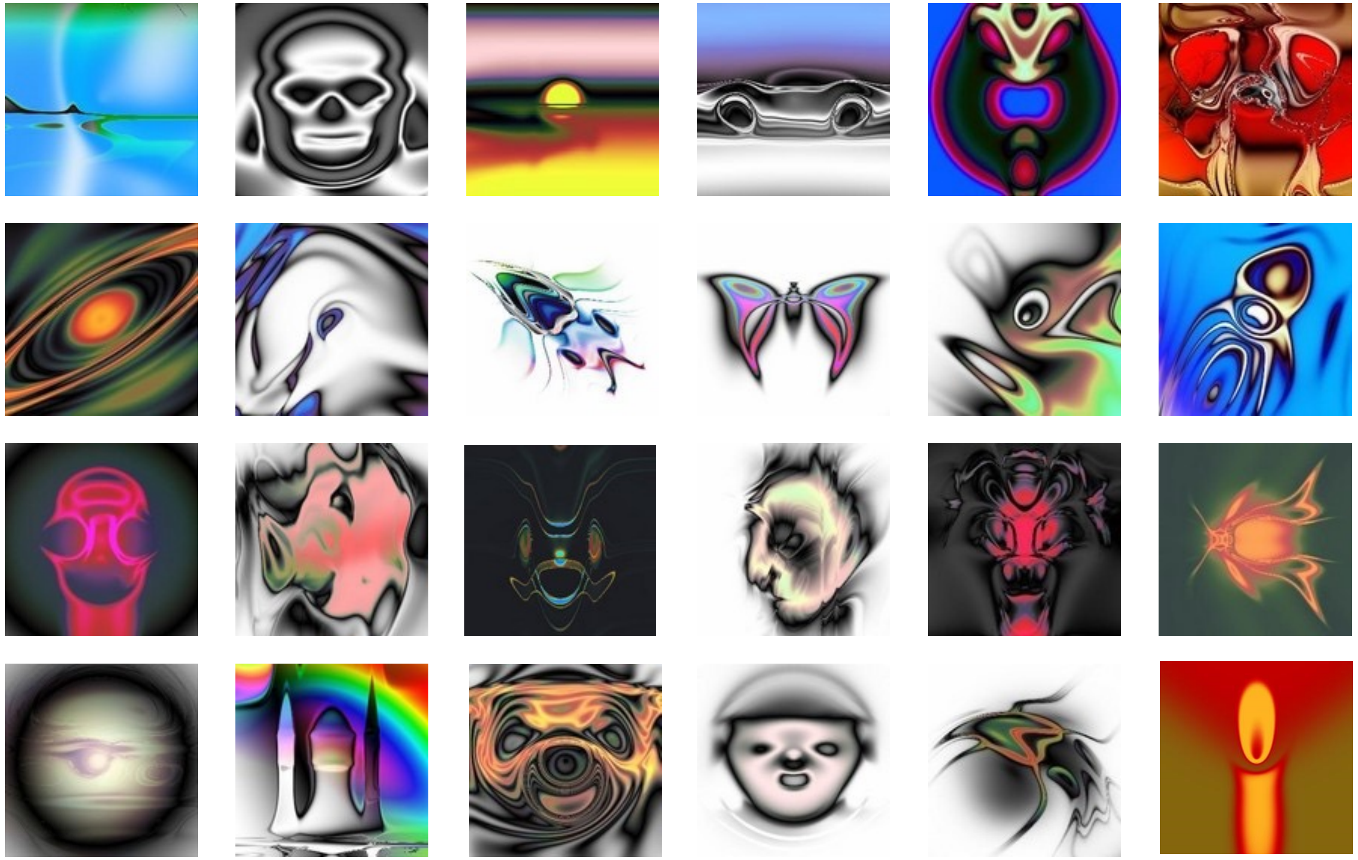
Compositional Pattern Producing Networks (CPPNs)

- CPPNs were devised by Stanley [2007] as an abstraction of development.
- A CPPN is a neural network that generates object properties as a function of position
- CPPN neurons can have a variety of activation functions suitable for geometric descriptions.
- CPPN produce symmetry, repetition, and repetition with variations, as observed in biological development



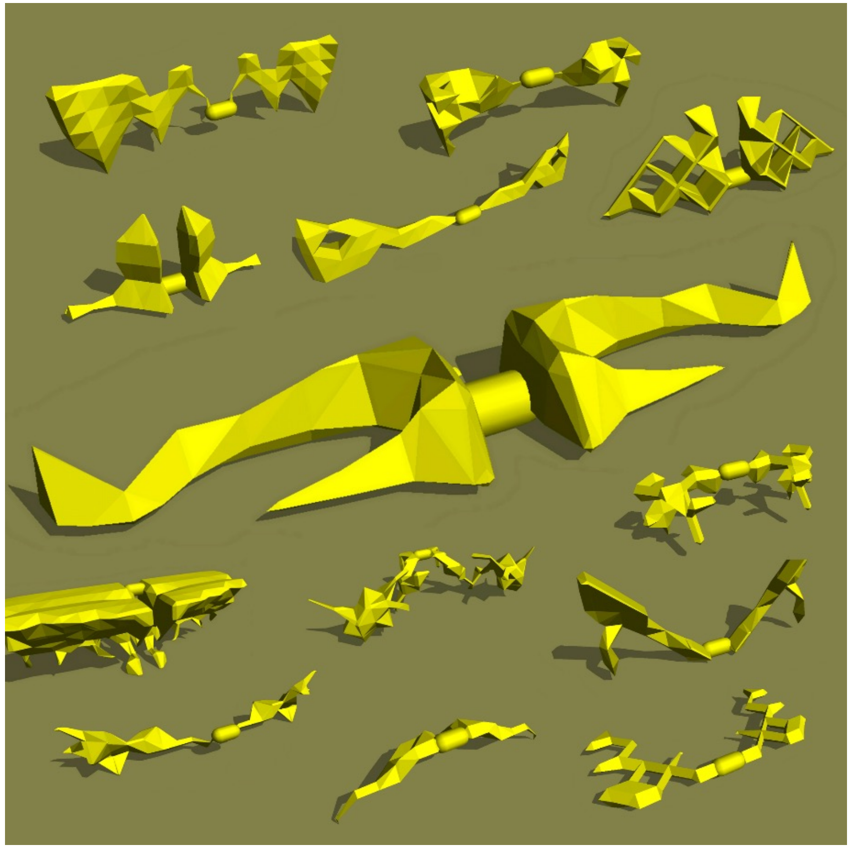


2-Dimensional images



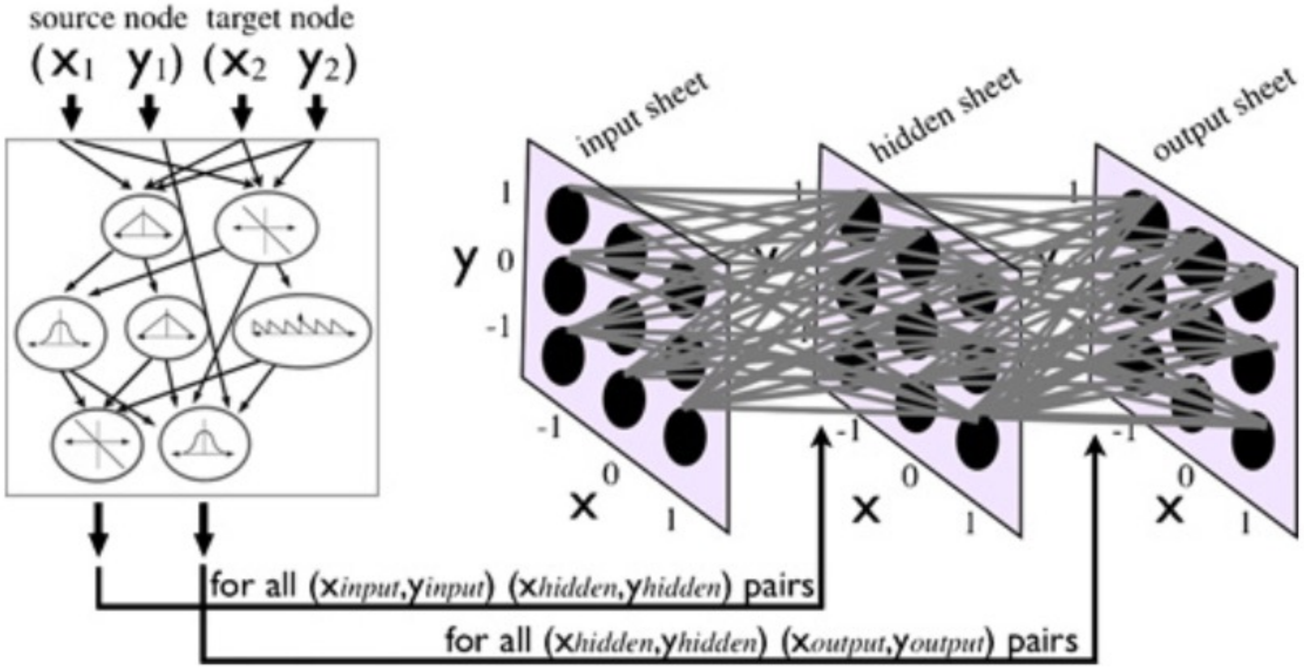
Picbreeder.org
[Secretan et al., 2007]

3-Dimensional objects



Robot morphologies
[Auerbach and Bongard, 2014]

Co-design of neural controllers and robotic bodies by CPPNs

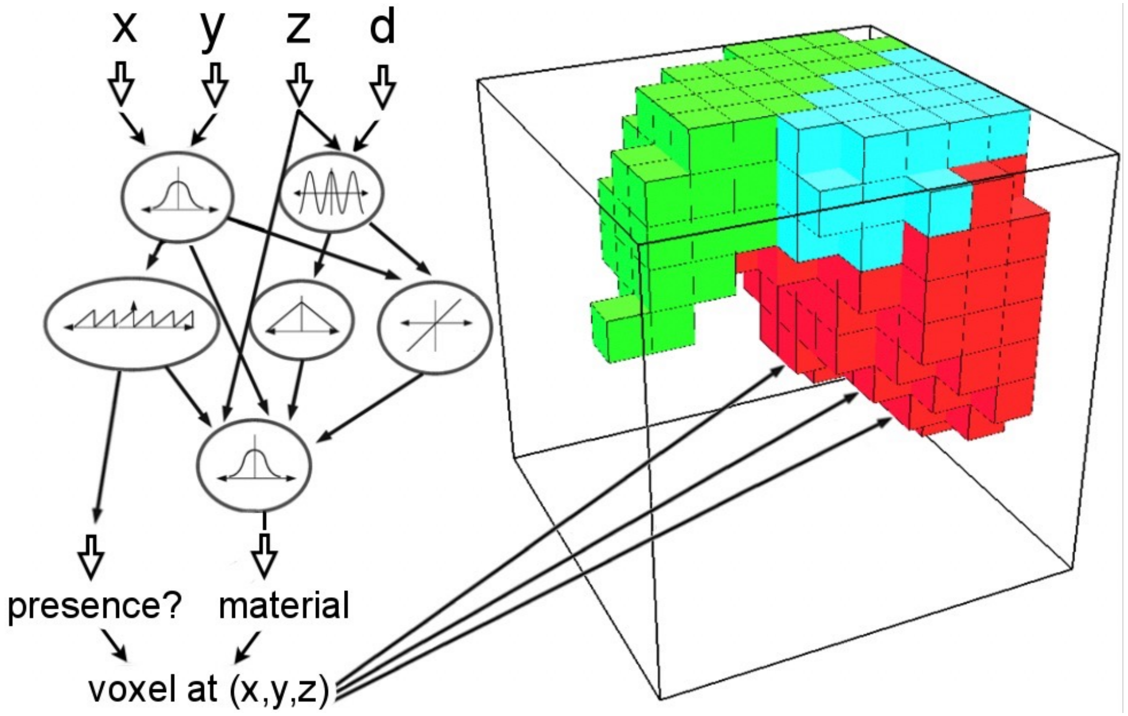


CPPNs can "paint" weights of neural network connections [Stanley et al., 2009], up to several million connections

CPPNs can be used to paint both the robot morphology and the weights of the neural controllers [Clune et al., 2013].

Encoding of soft-bodied robots

Cheney, MacCurdy, Clune, Lipson, 2013

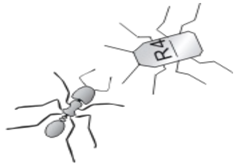


Green voxels undergo periodic volumetric actuations of 20%

Red voxels behave similarly to green ones, but with counter-phase actuation

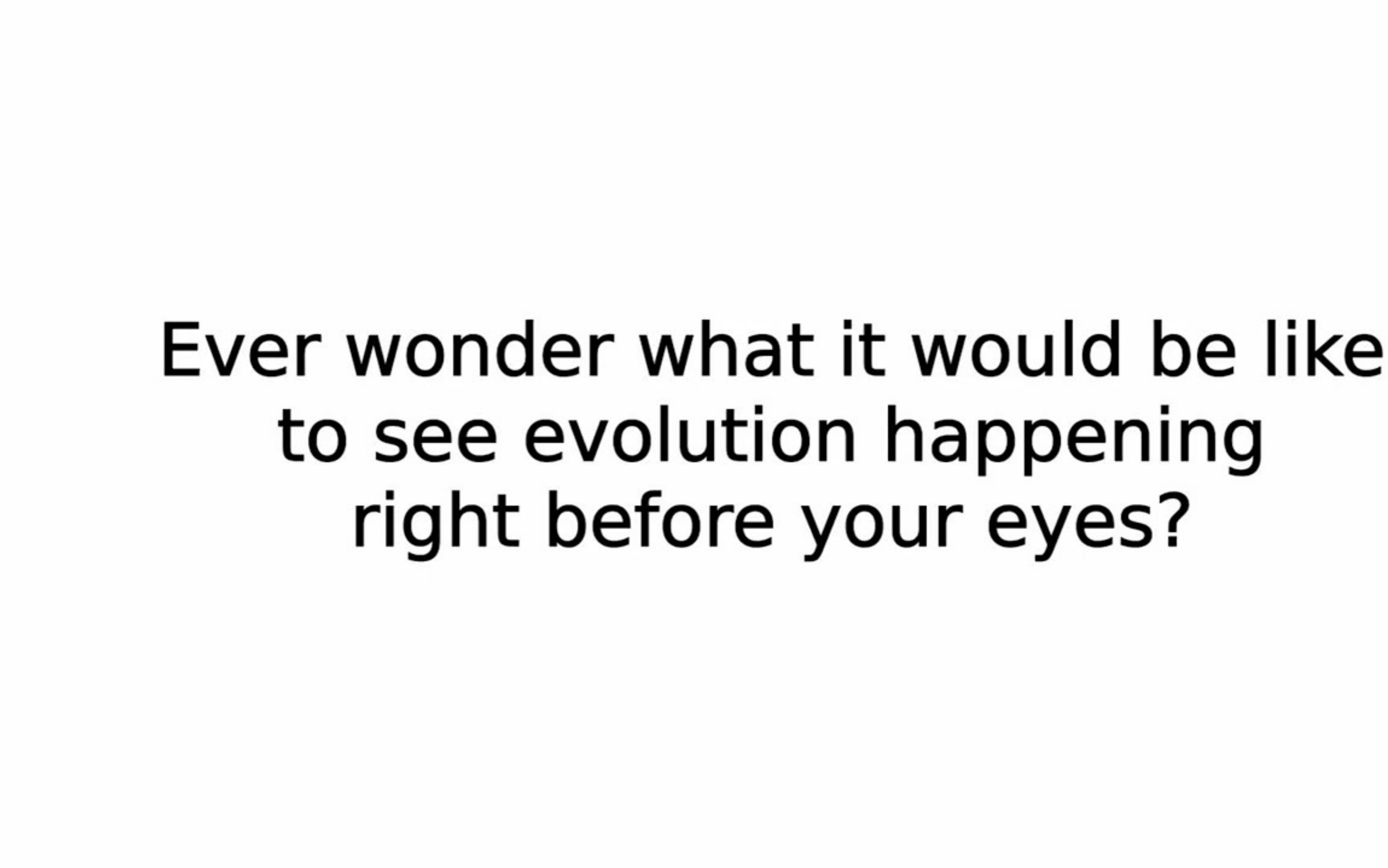
Light blue voxels are soft and passive, having no intrinsic actuation

Dark blue voxels are also passive, but are stiffer



Evolution of soft-bodied robots

Cheney, MacCurdy, Clune, Lipson, 2013

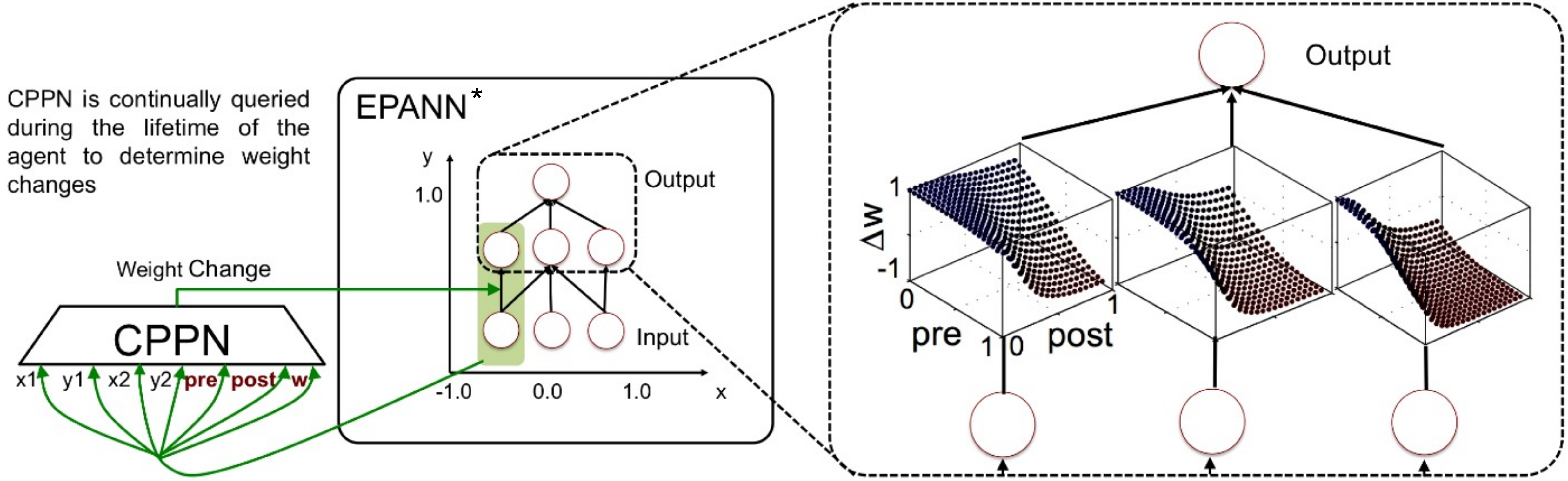


http://jeffclune.com/videos.html

Using CPPNs as learning rules

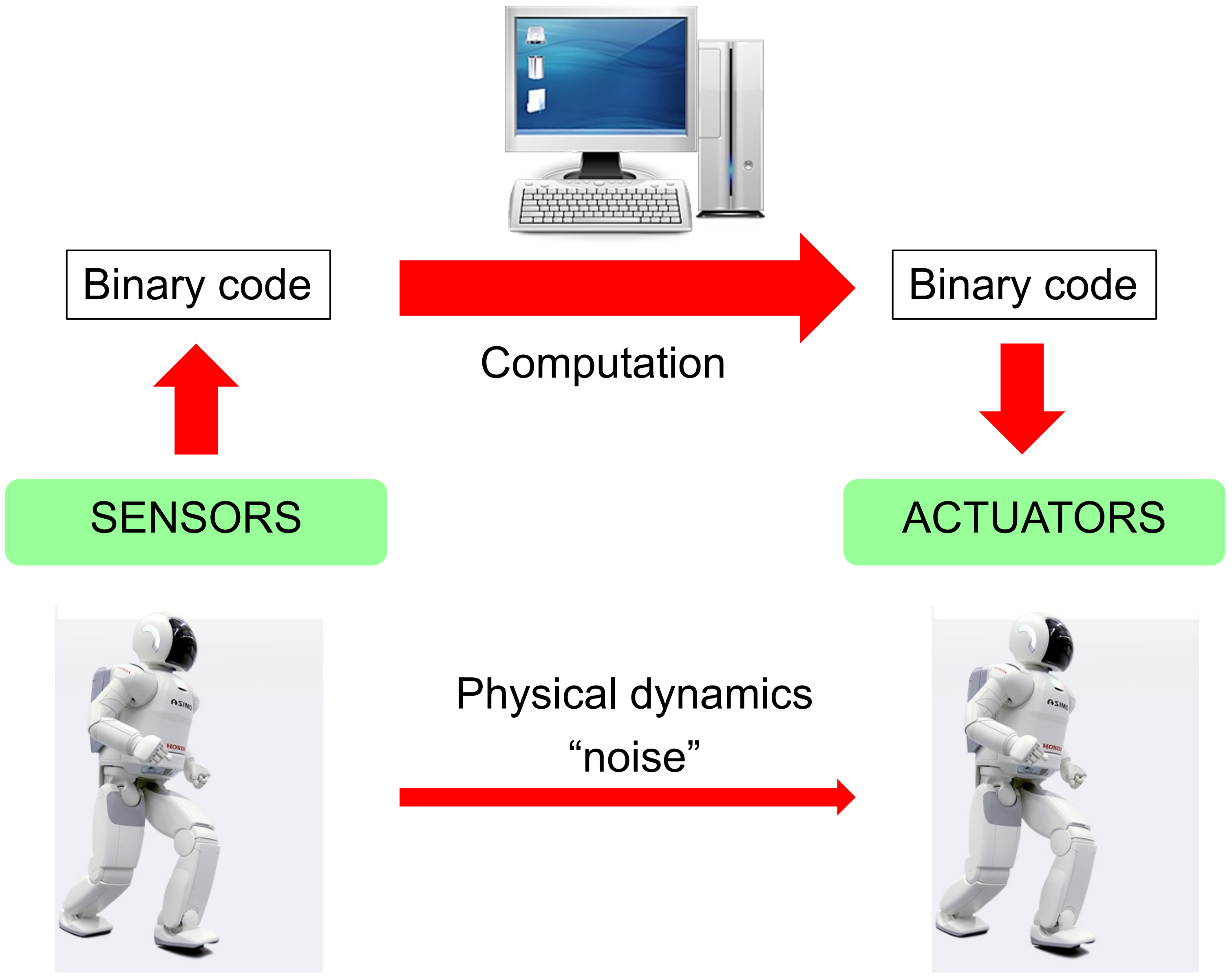
Risi and Stanley, 2010, 2014

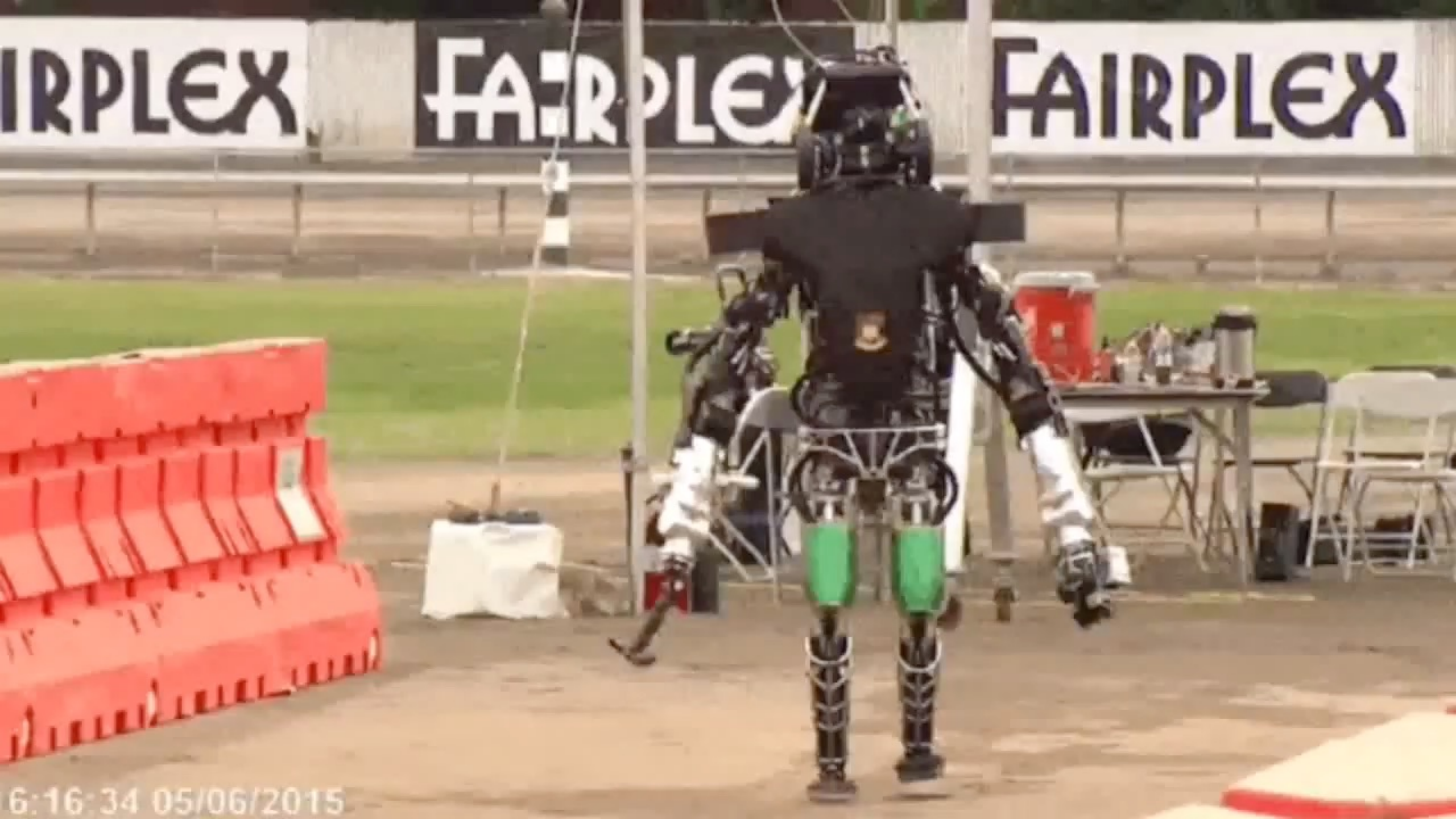
- Genetically encode and evolve the weights of the CPNN
- Use CPNN to compute weight updates of the neural controller at each time step of the robot lifetime
- Use robot's performance to compute fitness of the CPNN for selection



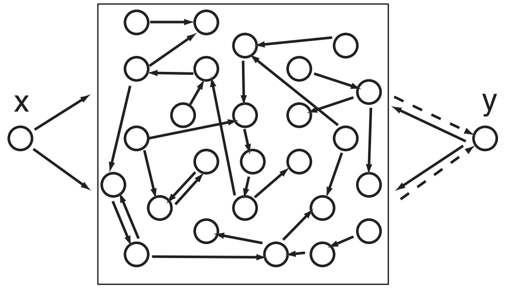
*Evolutionary Plastic Artificial Neural Network

Conventional Control




IRPLEX
FAIRPLEX
FAIRPLEX
6:16:34 05/06/2015

Morphological Computation



Binary code → Computation → Binary code

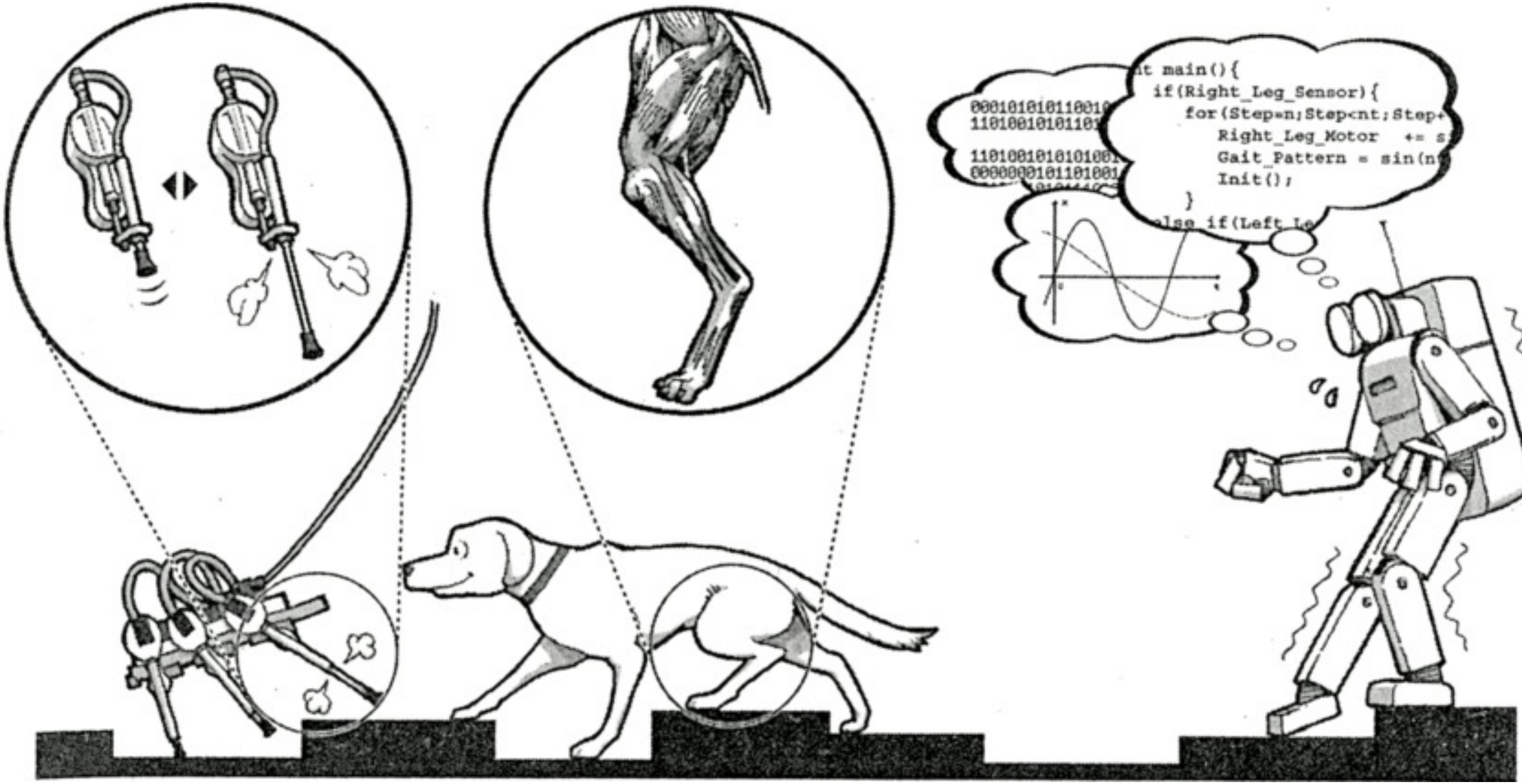
SENSORS

ACTUATORS

Morphology, dynamics, interaction →

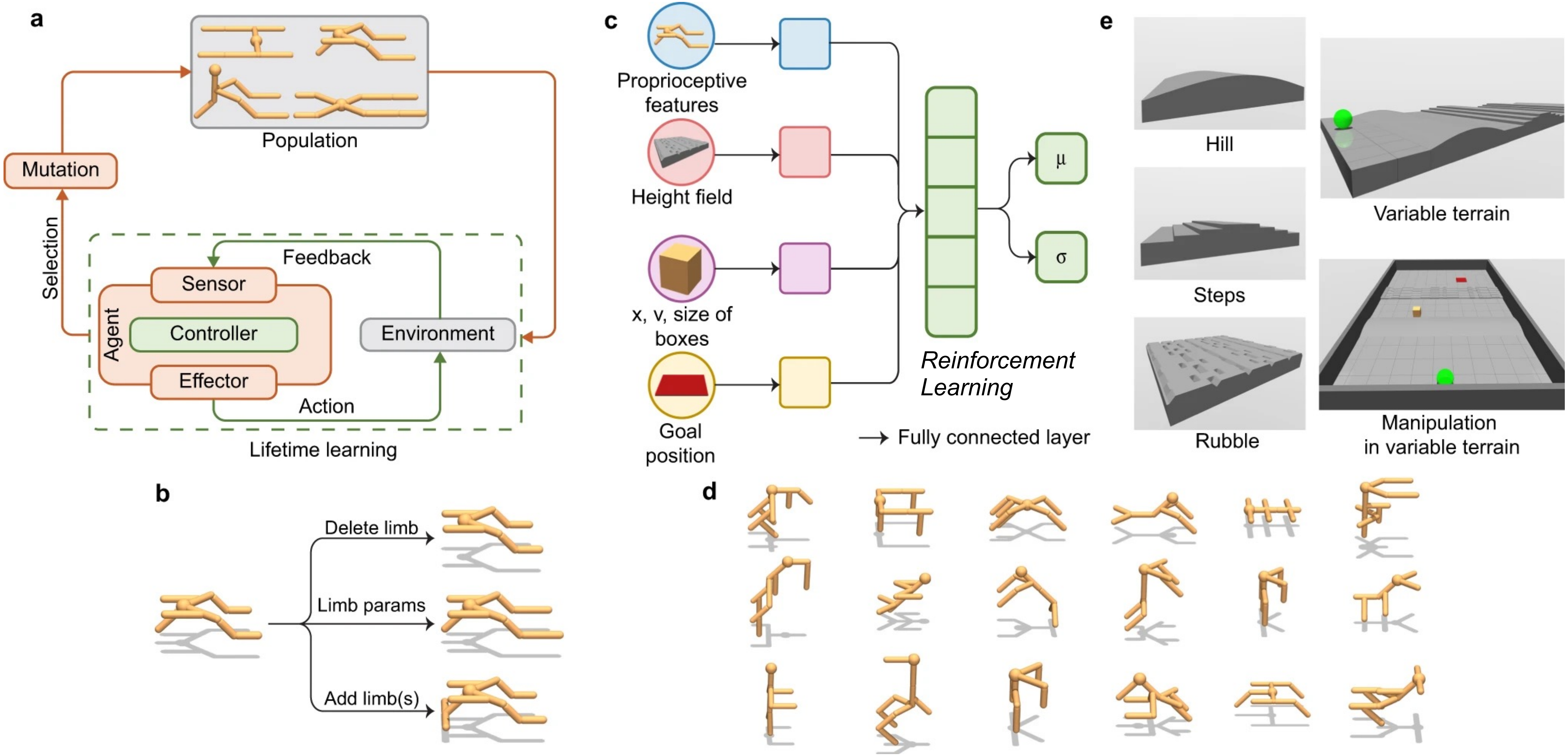
Pfeifer, Rolf, Max Lungarella, and Fumiya Iida (2007) Self-Organization, Embodiment, and Biologically Inspired Robotics. *Science* 318(5853), 1088–93.

Morphological computation simplifies control




BostonDynamics
Boston Dynamics
Boston Dynamics

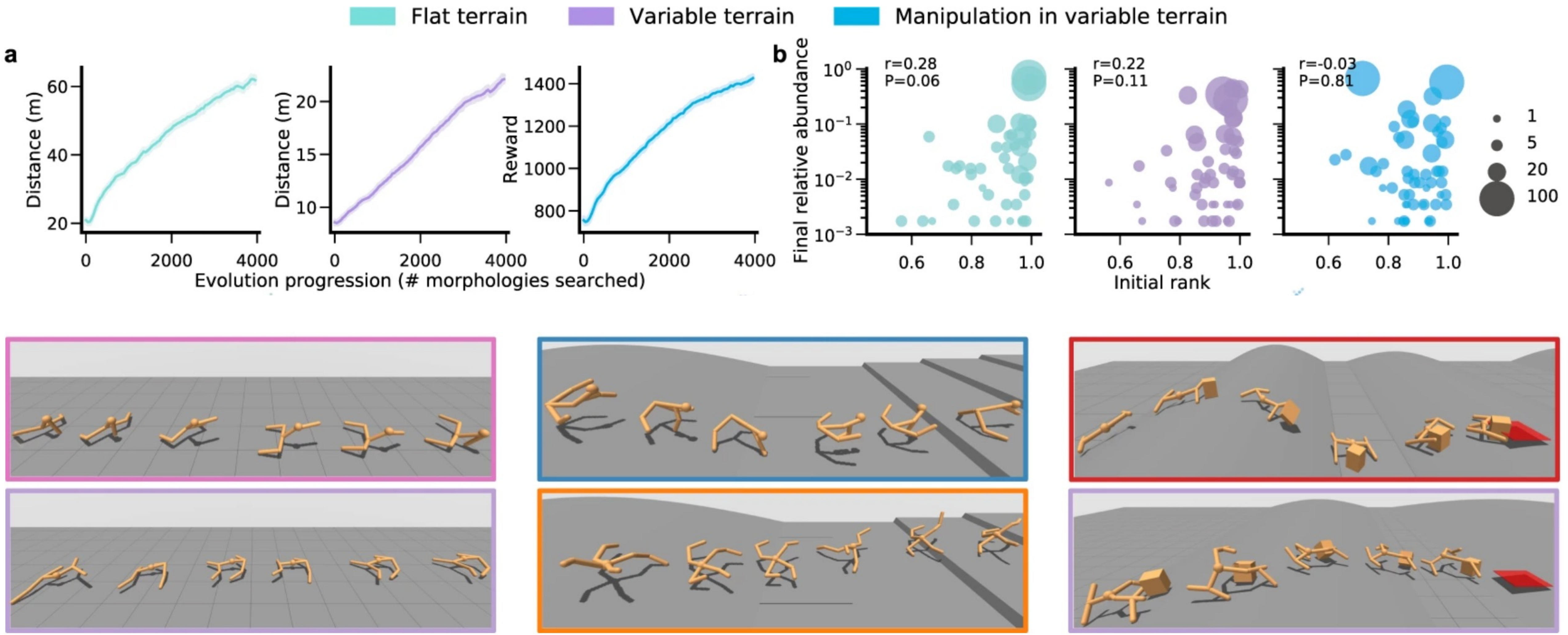
Morphological evolution of learning robots



Gupta A, Savarese S, Ganguli S, Fei-Fei L (2021) Embodied Intelligence via Learning and Evolution. *Nature Communications* 12(1), 5721

Local tournament selection preserves diversity

Population spread across 100’s of CPU, each simulating 4 individuals and reproducing the best one



Better bodies learn faster and better