Goals

- Develop a simulator for modular robots
- Develop a script to build the robots
  - Easy to read and edit by the user
  - Allows evolved robots to be saved, inspected and modified
- Implement a genetic algorithm (GA)
  - Evolve locomotion
  - Test the simulator

Modular robotics

- Motivations
  - Versatility
  - Robustness
  - Low cost?
- Applications
  - Search and rescue
  - Space exploration
  - Battlefield reconnaissance

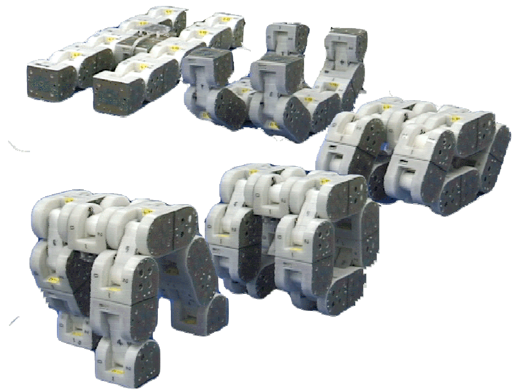
Co-evolving morphology and control

- Difficulties
  - Testing
  - Transfer from simulated to real world
- Promises
  - Evolve complex systems
  - Fitter individuals
  - Well adapted for modular robotics

Simulation

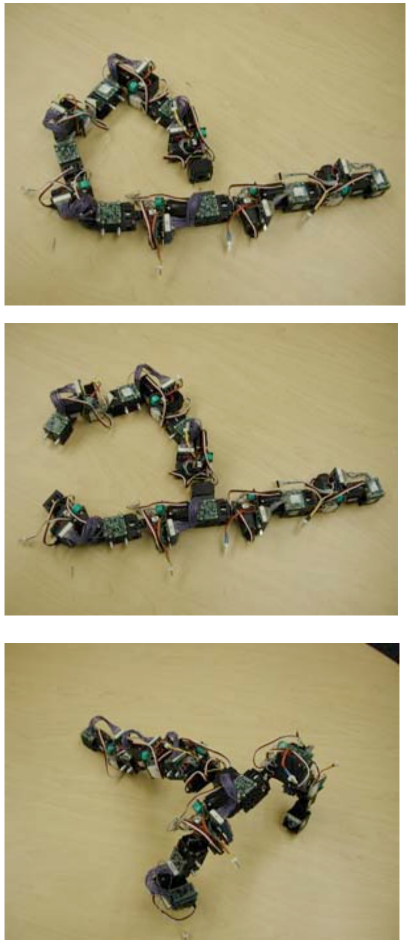
- Advantages
  - Speed
  - Low cost
- Closing the ‘reality gap’
  - Add noise
  - Sampling
  - Minimal simulation

State-of-the-art



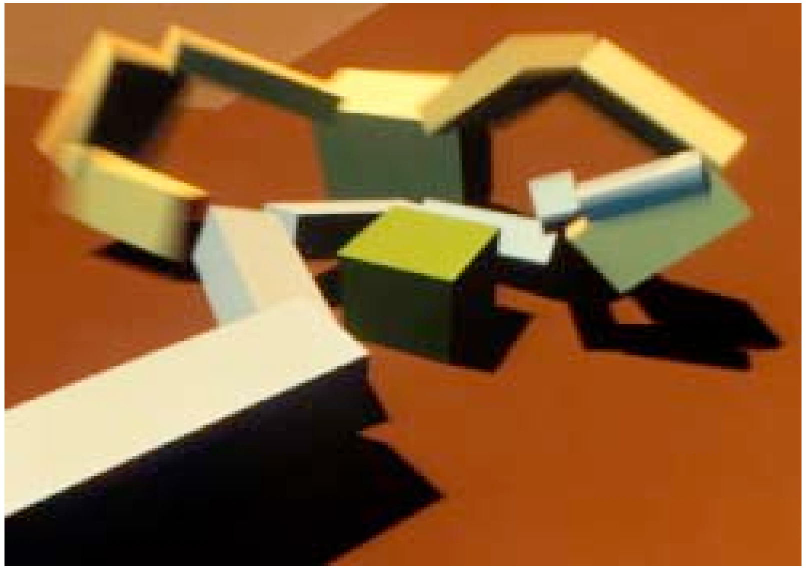
Modular Transformer
(AIST)

PolyBot (PARC)



CONRO (USC)

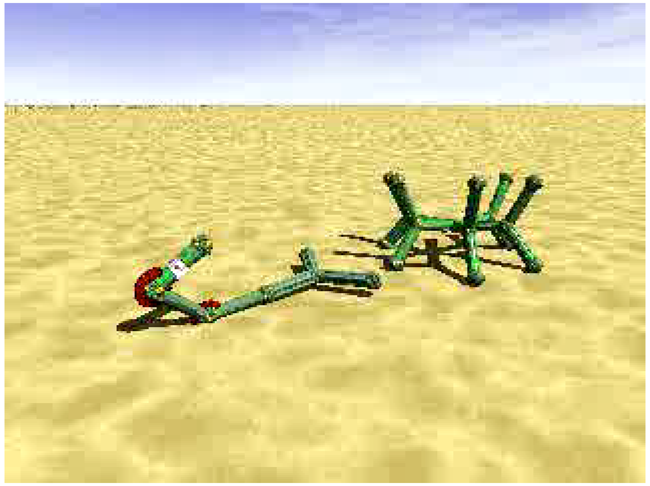
Karl Sims' block creatures



- Co-evolution of morphology and neural network
- Competition
- Genotype is directed graph
- Very similar to Adam

Framsticks

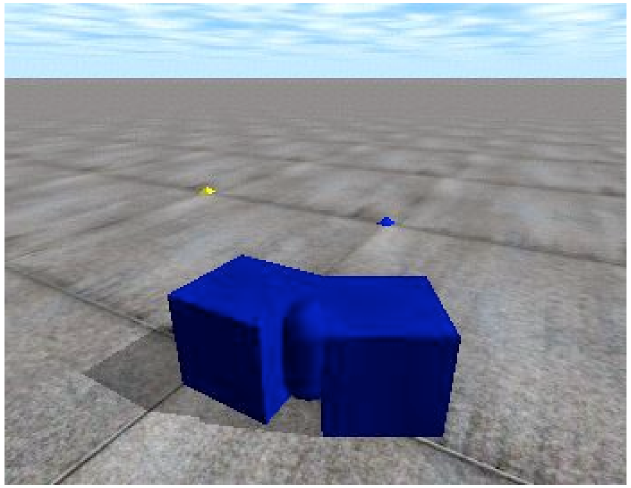
- Artificial life
- Nice user interface
- Various genotypes
- Many parameters of the GA can be set by the user
- Very similar to Adam



Adam - overview

- Modular robots
  - No self-reconfiguration
  - No cycles
  - Homogenous
- Simulation
  - Implemented with ODE
  - Rigid body dynamics (kinematics, friction, collision etc)
  - Simulation world: Infinite plane

Hinge module



- Other modules can be attached at every position
- Rigid, powered or elastic

Hinge parameters

- Initial angle
- Low and high stop
- For powered hinges:
  - Maximal force of the motor
  - Control: Amplitude, frequency and phase
    $\theta = A \sin(2\pi f + \varphi)$
- For elastic hinges
  - Elasticity and damping constants

Script - overview

- Formally defined with a lexical and a syntactic grammar
- Structural part
  - Defines how modules are attached to each other
  - Each module is given a unique identifier
- Parameters are set in the second part
  - Set default parameters
  - Set parameters of specific modules

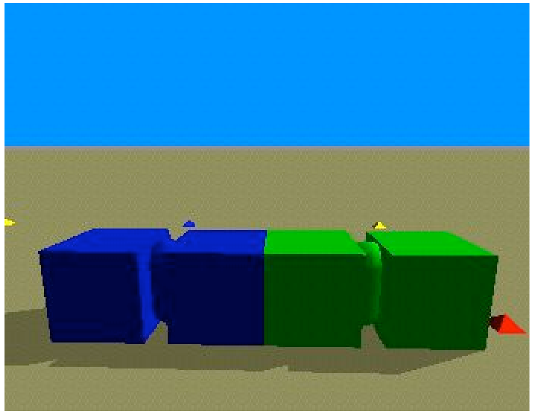
Script – defining structures

- Sequential building plan
- The first module is the head of the robot
- Add new modules. Define:
  - Where?
  - With which position?
  - With which orientation?

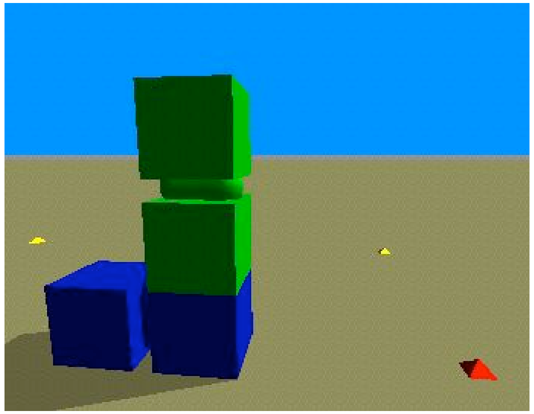
Definition: Positions of a hinge

P0: First cube back face
P1: First cube top face
P2: First cube right face
P3: First cube bottom face
P4: First cube left face
P5: Second cube top face
P6: Second cube right face
P7: Second cube bottom face
P8: Second cube left face
P9: Second cube front face

Attaching limbs (1)

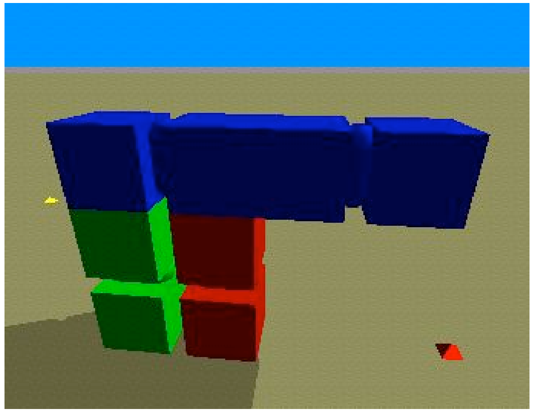


Ha Hb

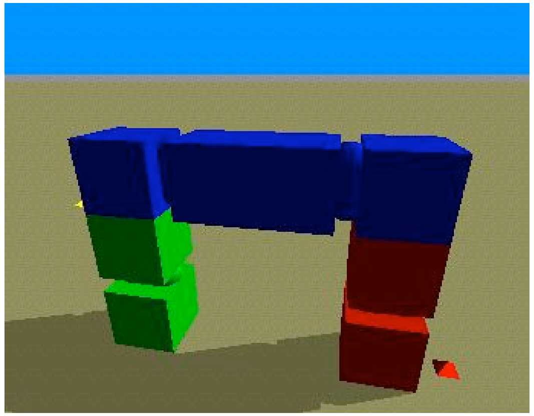


Ha P5(Hb)

Attaching limbs (2)

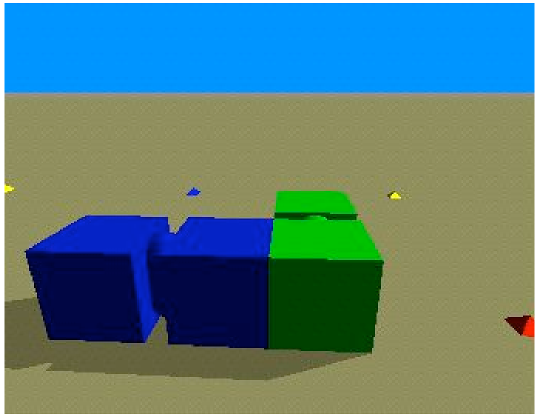


Ha P3(Hb) P7(Hc) Hd

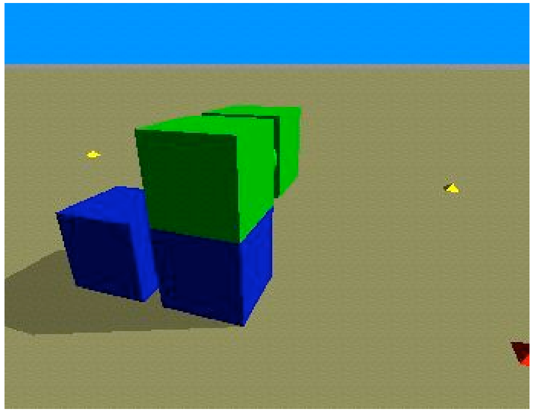


Ha P3(Hb) Hd P7(Hc)

Specifying the position a hinge gets attached with



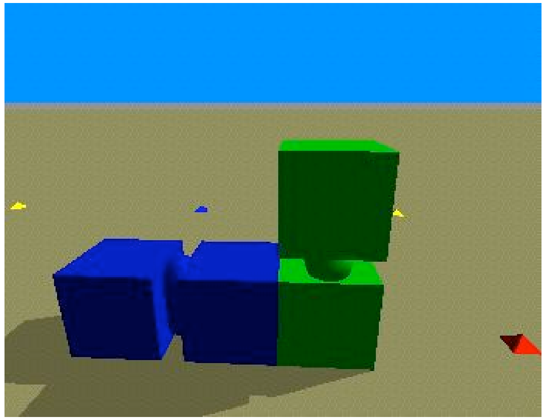
Ha P4 Hb



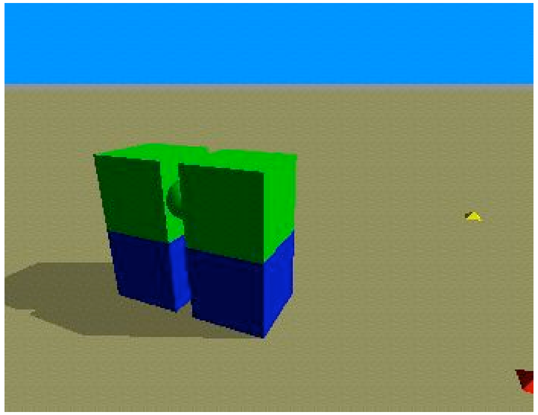
Ha P5 (P4 Hb)

BIOLOGICALLY INSPIRED ROBOTICS GROUP (BIRG)

Specifying the orientation



Ha P4 E Hb



Ha P5 (P4 E Hb)

Defaults

- The default position to attach a limb is P9
- The default position with which a hinge is attached to another one is P0
- The default orientation is North
- Therefore:

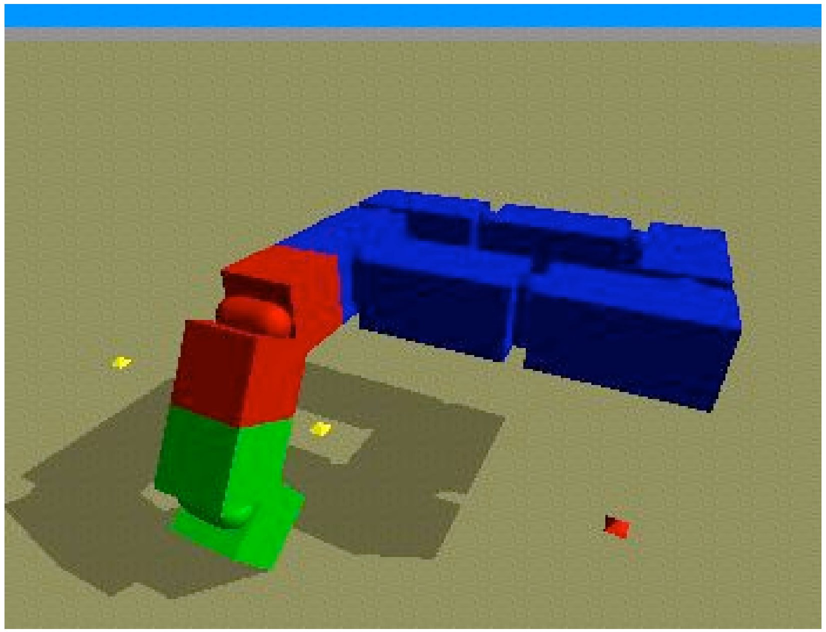
  Ha Hb = Ha P9(Hb) = Ha P0 Hb = Ha N Hb
  = Ha P9 (P0 N Hb) = …

Setting parameters

- Modules have default parameters
- The defaults can be reset by the user
- Notation:

  identifier.function(arguments)
- Hinge parameter-setting functions:
  - *initAngle(□)*
  - *powered(isPowered, loStop, hiStop, Fmax, A, f, □)*
  - *soft(isSoft, elast, damp)*

Example (1)



STRUCTURE

H_body0

P2(E H_leg H_foot)

H_body1 P4 H_body2

P4 H_body3 H_body4

P4 H_body5

PARAMETERS

H_leg.initAngle(-60)

H_foot.initAngle(-60)

H_leg.powered(true,
-60, 60, 100, 60, 0.2, 0)

H_foot.soft(true, 50, 0)

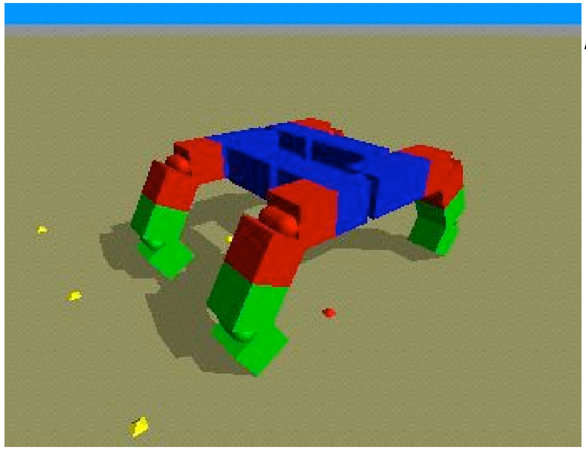
Example (2)

Adam

BIOLOGICALLY INSPIRED ROBOTICS GROUP (BIRG)

```
//////////// LEG ////////////////
leg {
   STRUCTURE
   H_leg H_foot

   PARAMETERS
   H.initAngle(-60)  // default
   H_leg.powered(true, -60, 60,
                 100, 60, 0.2, 0)
   H_foot.soft(true, 50, 0)
}
…
```

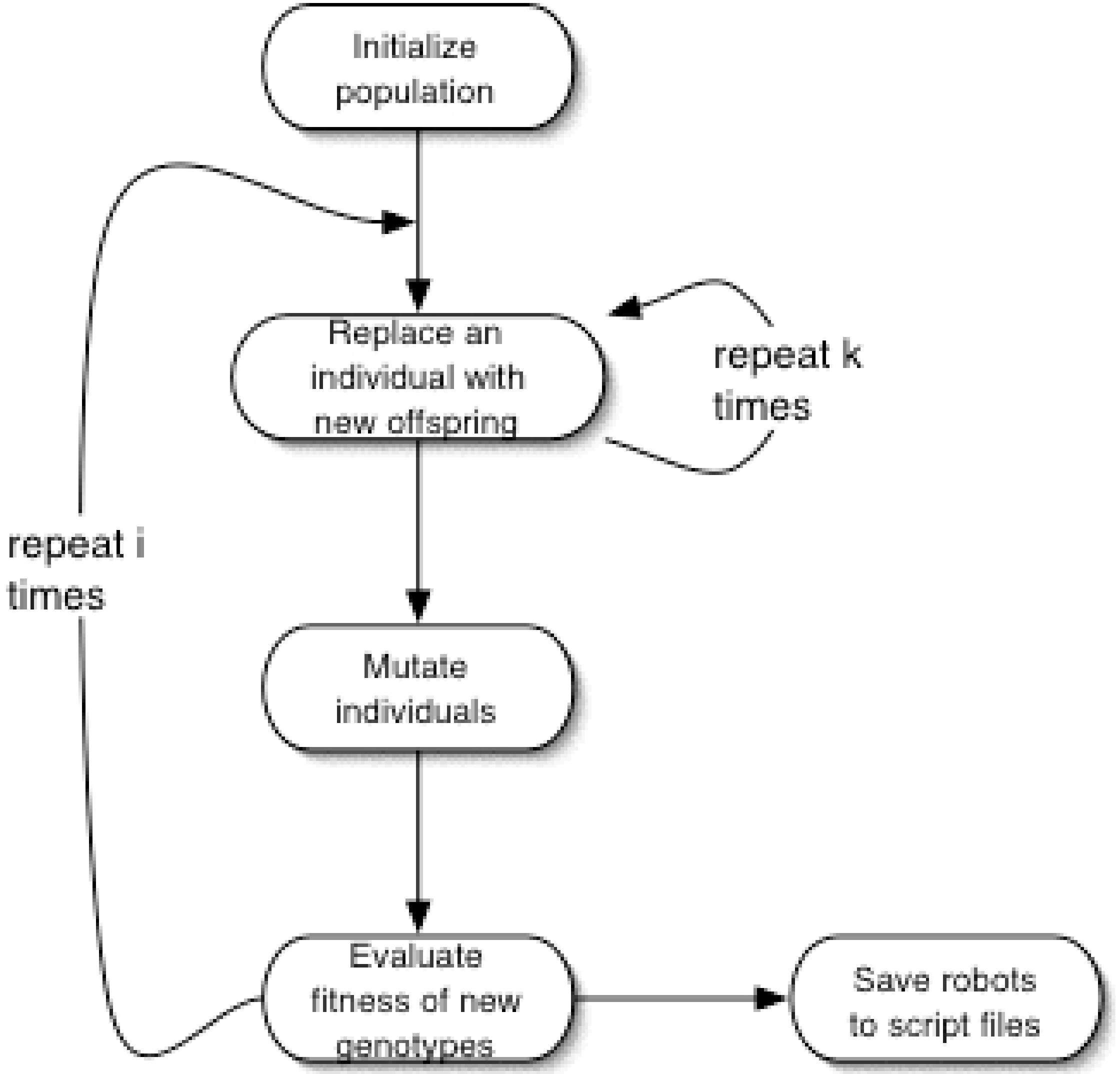


```
/////////// BODY ///////////////
STRUCTURE
H_body0
   P2(E leg)
H_body1 P4 H_body2
   P0(W leg)
P4 H_body3
   P2(E leg)
H_body4 P4 H_body5
   P0(W leg)

PARAMETERS
```

Genetic algorithm Adam

BIOLOGICALLY INSPIRED ROBOTICS GROUP (BIRG)



Phenotype space

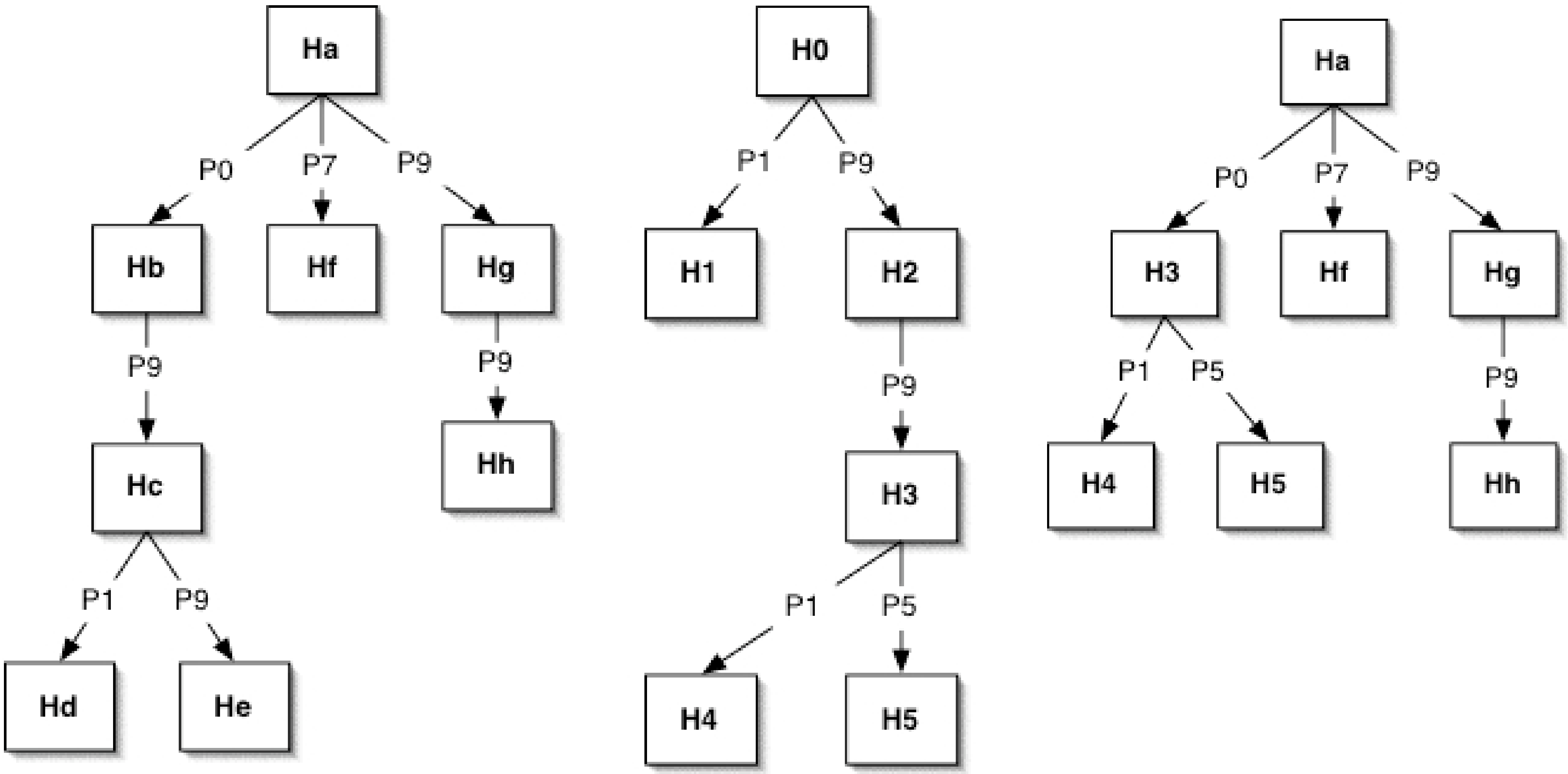
1. There are an infinite number of modules available of each type.
2. There's a finite number N of module types used (usually few).
3. There's infinite space available to build the robot (there are no limitations on size and form of the robot).

The Adam phenotype space consists of all robots in the simulated world that could be built theoretically in the real world with corresponding hardware modules under hypothesis 1-3.

Genetic encoding

- The script is not a good choice
- The phenotype space is structured
  - Genetically
  - With respect to fitness values
  - Goal: Find a genetic encoding that correlates the two
- Developmental encodings
  - Better structured individuals
  - Fitter individuals?
- Adam uses Trees

Crossover


Ha
P0
P7
P9
Hb
Hf
Hg
P9
P9
Hc
Hh
P1
P9
Hd
He
H0
P1
P9
H1
H2
P9
H3
P1
P5
H4
H5
Ha
P0
P7
P9
H3
Hf
Hg
P1
P5
P9
H4
H5
Hh

Mutation

- Acts on all parameters and on the structure
- Sub trees can be deleted
- Modules can be added
- Position and orientation of attachment might change

Initialization

- Default positions have higher probability
  - Increases probability of building a legal structure
- Parameters are set ‘reasonable’
  - Frequency constant
  - Low stop = high stop
  - Phase is a multiple of □/6
  - etc
- Reinitialize illegal robots

Selection and replacement

- Rank-proportional roulette wheel method
- Probability of an individual make offspring:

  $p_s(i) = (N + 1 - r(i)) / \sum r(i)$
- Probability of an individual to be deleted:

  $p_r(i) = (r(i)-1) / (\sum r(i)-1)$
- Steady-state evolution

Goals

- Develop a simulator for modular robots
- Develop a script to build the robots
  - Easy to read and edit by the user
  - Allows evolved robots to be saved, inspected and modified
- Implement a genetic algorithm (GA)
  - Evolve locomotion
  - Test the simulator

Goals

- Develop a simulator for modular robots
- Develop a script to build the robots
  - Easy to read and edit by the user
  - Allows evolved robots to be saved, inspected and modified
- Implement a genetic algorithm (GA)
  - Evolve locomotion
  - Test the simulator

Goals

- Develop a simulator for modular robots
- Develop a script to build the robots
  - Easy to read and edit by the user
  - Allows evolved robots to be saved, inspected and modified
- Implement a genetic algorithm (GA)
  - Evolve locomotion
  - Test the simulator