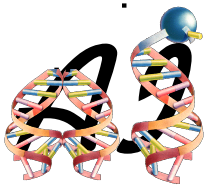
Genetic Programming and Lindenmayer Systems

Christian Jacob
Department of Computer Science
University of Calgary

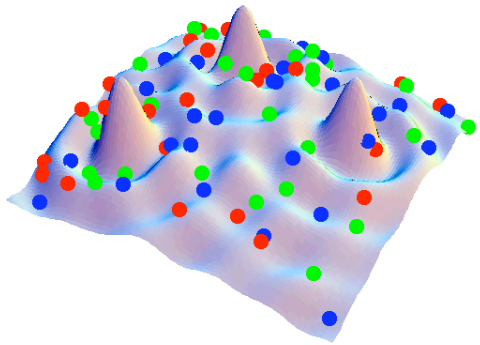
CPSC 601.73 — Winter 2003

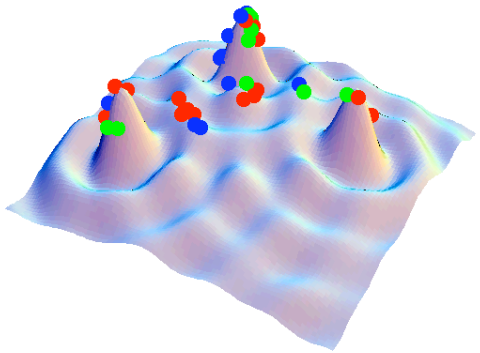


Genetic Algorithms (GA)

John H. Holland
(1975)
David Goldberg
(1989)

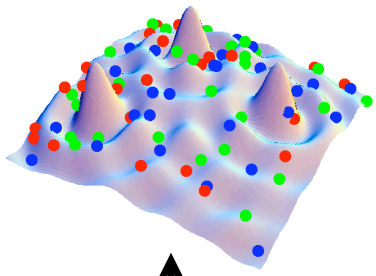
Evolutionary Optimization





- Knowledge Reservoir
  Set of possible solutions
  - *Gleaning a reservoir of knowledge from interactions with the environment.*
- Selection
  Fitness-dependent number of offspring
  - *The sieve of selection culls out incorrect / unuseful "knowledge".*
- Variation
  Variations of individual solutions
  - *The learning system invents new variants of its old ideas that are tested against environmental demands.*

How Genetic Algorithms Work

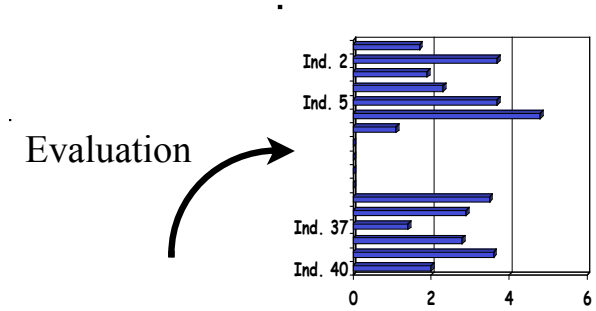


Interpretation

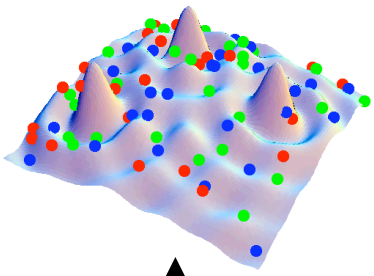
{1,0,1,1,0,1,0,0,1,0,1,1}
{0,1,1,1,1,0,0,1,0,0,0,1}
{1,1,0,0,0,1,0,1,0,1,0,0}
...

{1,0,1,0,0,1,1,1,0,1,1,1}
{0,0,1,1,0,1,1,1,0,1,0,0}
{1,0,0,1,0,1,1,1,0,0,0,1}

How Genetic Algorithms Work



Evaluation

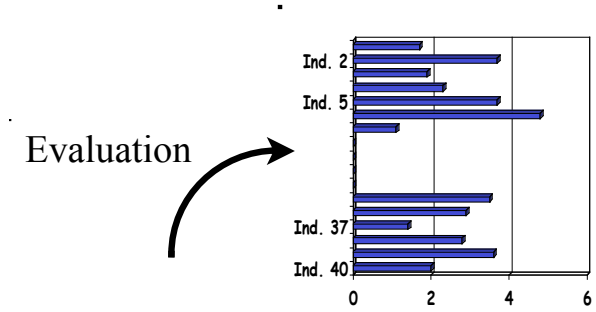


Interpretation

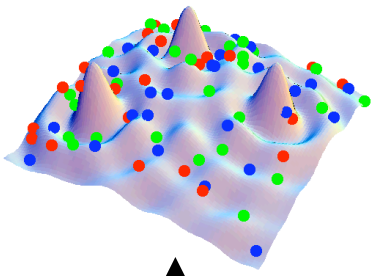
{1,0,1,1,0,1,0,0,1,0,1,1}
{0,1,1,1,1,0,0,1,0,0,0,1}
{1,1,0,0,0,1,0,1,0,1,0,0}
...

{1,0,1,0,0,1,1,1,0,1,1,1}
{0,0,1,1,0,1,1,1,0,1,0,0}
{1,0,0,1,0,1,1,1,0,0,0,1}

How Genetic Algorithms Work



Evaluation



Interpretation

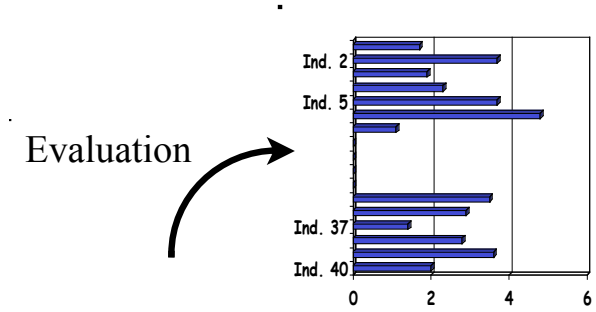
{1,0,1,1,0,1,0,0,1,0,1,1}
{0,1,1,1,1,0,0,1,0,0,0,1}
{1,1,0,0,0,1,0,1,0,1,0,0}
...

{1,0,1,0,0,1,1,1,0,1,1,1}
{0,0,1,1,0,1,1,1,0,1,0,0}
{1,0,0,1,0,1,1,1,0,0,0,1}

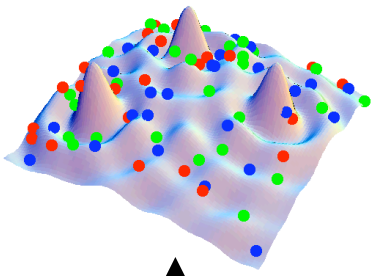
Selection

{0,0,1,1,0,1,1,1,0,1,0,0} {1,1,0,0,0,1,0,1,0,1,0,0}

How Genetic Algorithms Work



Evaluation



Interpretation

{1,0,1,1,0,1,0,0,1,0,1,1}
{0,1,1,1,1,0,0,1,0,0,0,1}
{1,1,0,0,0,1,0,1,0,1,0,0}
...

{1,0,1,0,0,1,1,1,0,1,1,1}
{0,0,1,1,0,1,1,1,0,1,0,0}
{1,0,0,1,0,1,1,1,0,0,0,1}

{1,0,1,1,0,1,0,0,1,0,1,1}
{0,1,1,1,1,0,0,1,0,0,0,1}
{1,1,0,0,0,1,0,1,0,1,0,0}
...

{1,0,1,0,0,1,1,1,0,1,1,1}
{0,0,1,1,0,1,1,1,0,1,0,0}
{1,0,0,1,0,1,1,1,0,0,0,1}

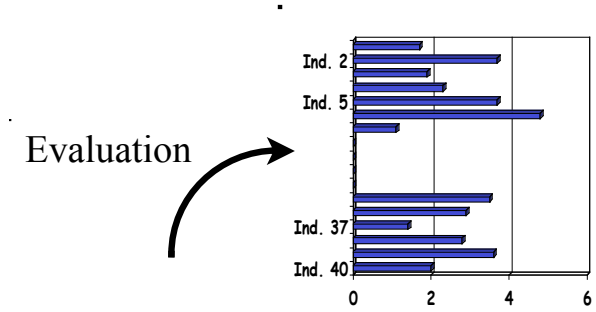
Selection

{0,**0**,1,1,0,**1**,1,1,0,1,**0**,0} {1,1,**0**,**0**,0,1,0,1,0,**1**,0,0}

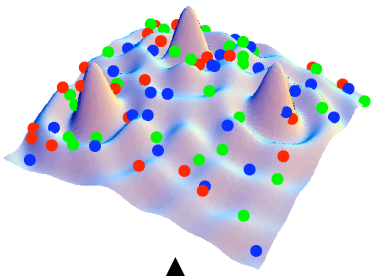
Mutation

{0,**1**,1,1,0,**0**,1,1,0,1,**1**,0} {1,1,**1**,**1**,0,1,0,1,0,**0**,0,0}

How Genetic Algorithms Work



Evaluation



Interpretation

{1,0,1,1,0,1,0,0,1,0,1,1}
{0,1,1,1,1,0,0,1,0,0,0,1}
{1,1,0,0,0,1,0,1,0,1,0,0}
...

{1,0,1,0,0,1,1,1,0,1,1,1}
{0,0,1,1,0,1,1,1,0,1,0,0}
{1,0,0,1,0,1,1,1,0,0,0,1}

{1,0,1,1,0,1,0,0,1,0,1,1}
{0,1,1,1,1,0,0,1,0,0,0,1}
{1,1,0,0,0,1,0,1,0,1,0,0}
...

Selection

{1,0,1,0,0,1,1,1,0,1,1,1}
{0,0,1,1,0,1,1,1,0,1,0,0}
{1,0,0,1,0,1,1,1,0,0,0,1}

{0,0,1,1,0,1,1,1,0,1,0,0} {1,1,0,0,0,1,0,1,0,1,0,0}

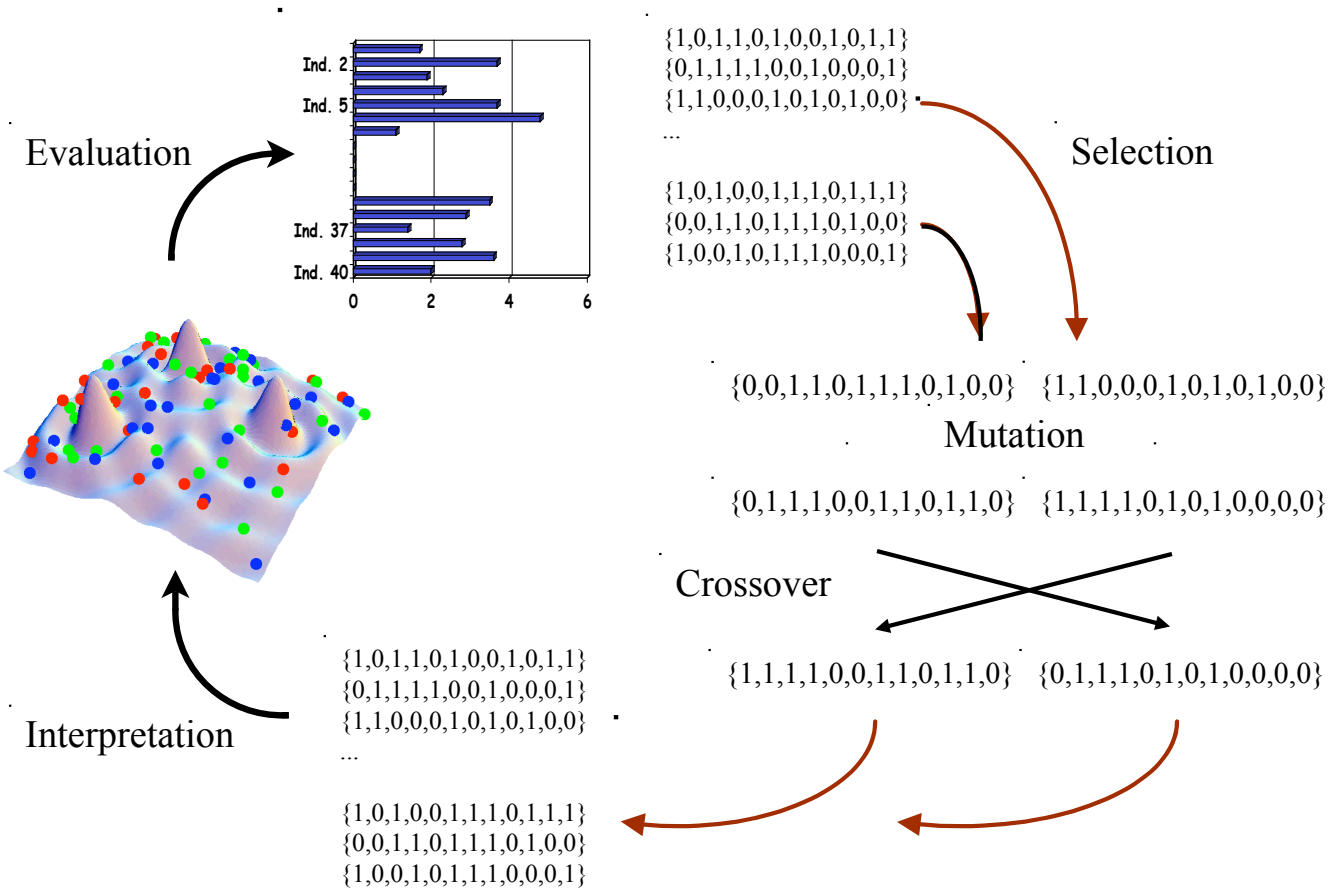
Mutation

{0,1,1,1,0, 0,1,1,0,1,1,0} {1,1,1,1, 0,1,0,1,0,0,0,0}

Crossover

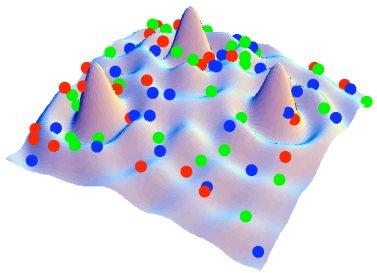
{1,1,1,1,0,0,1,1,0,1,1,0} {0,1,1,1,0,1,0,1,0,0,0,0}

How Genetic Algorithms Work

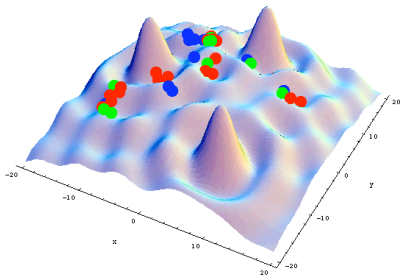


Evolution with Varying Objective Function

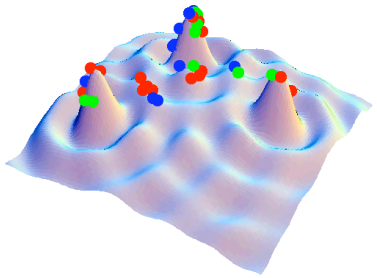
Generation 0



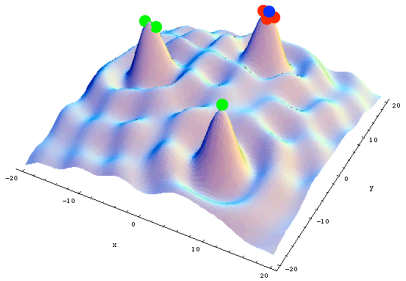
Generation 11



Generation 10



Generation 30



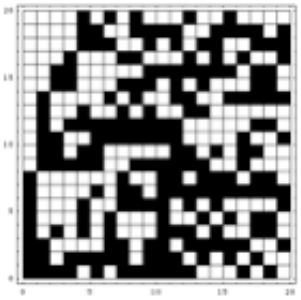
Evolution with Varying Objective Function

Generation 0 Generation 10

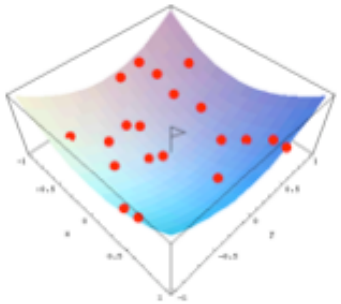
Evolution with Varying Objective Function

Generation 11 Generation 30

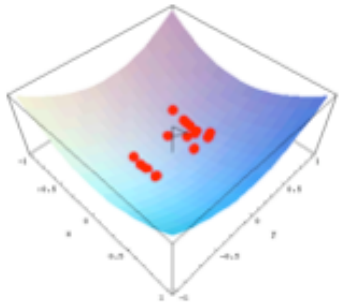
Comparison of Genetic Operators (Genotypes)



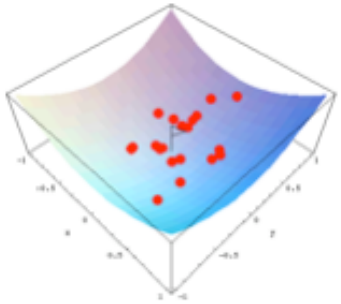
Initial Population



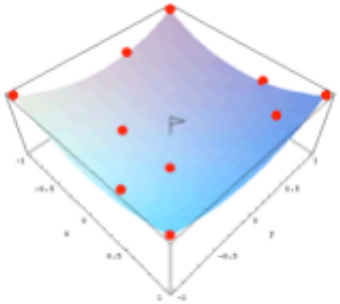
Mutation



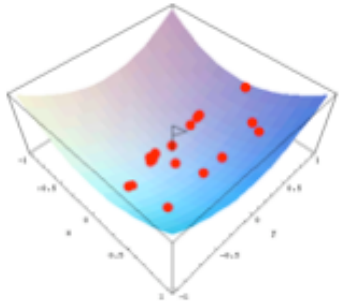
Recombination



Inversion

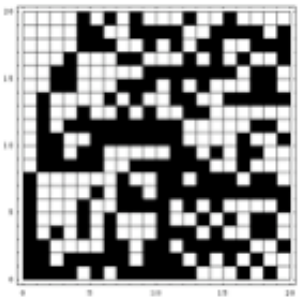


Deletion

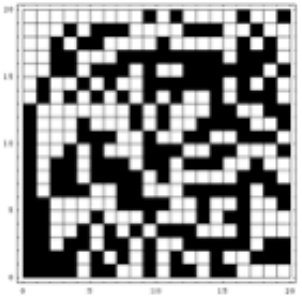


Duplication

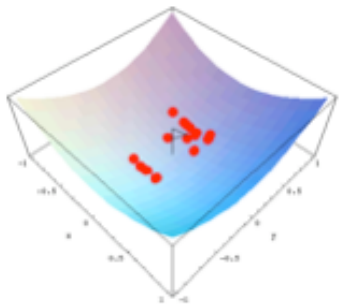
Comparison of Genetic Operators (Genotypes)



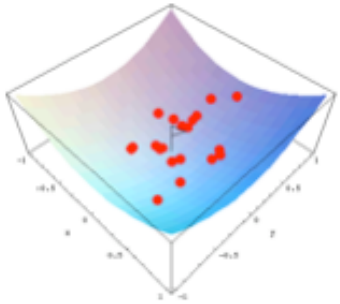
Initial Population



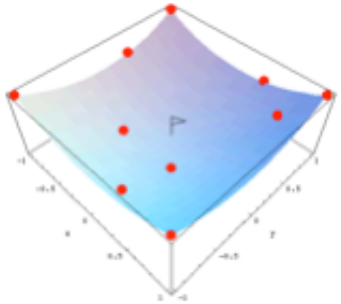
Mutation



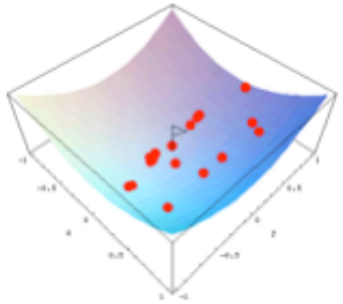
Recombination



Inversion

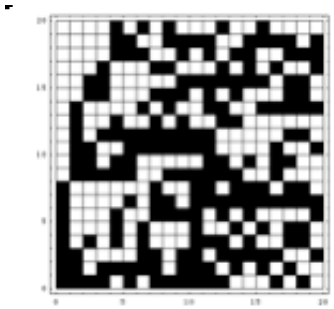


Deletion

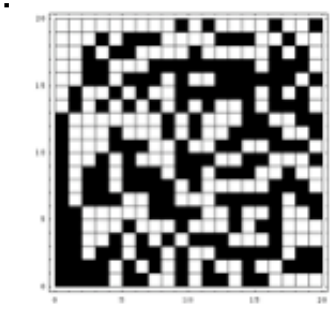


Duplication

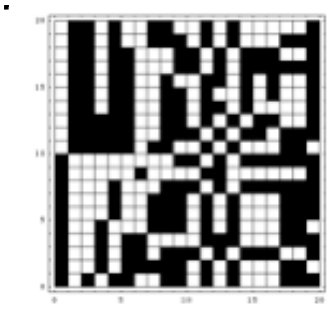
Comparison of Genetic Operators (Genotypes)



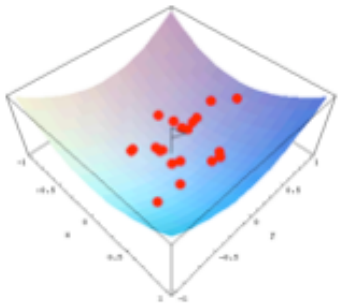
Initial Population



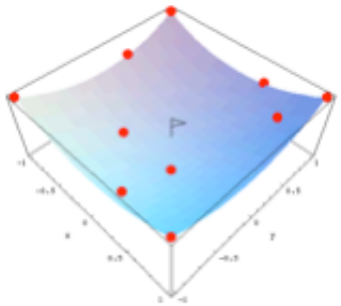
Mutation



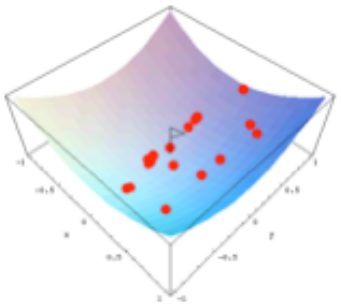
Recombination



Inversion

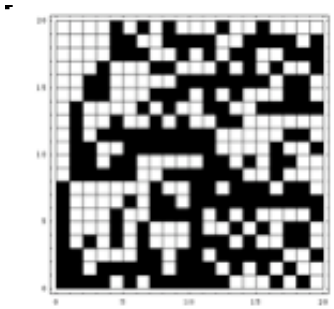


Deletion

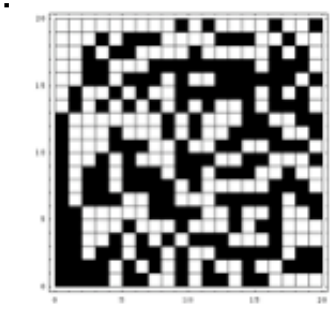


Duplication

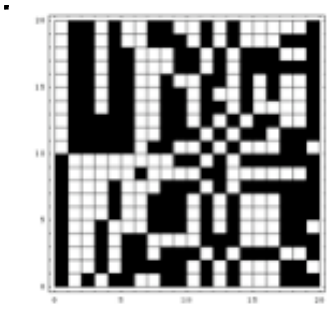
Comparison of Genetic Operators (Genotypes)



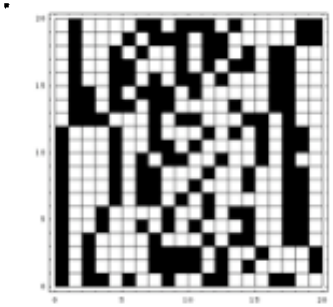
Initial Population



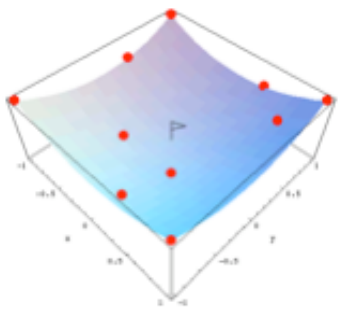
Mutation



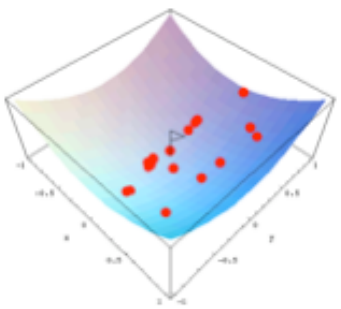
Recombination



Inversion

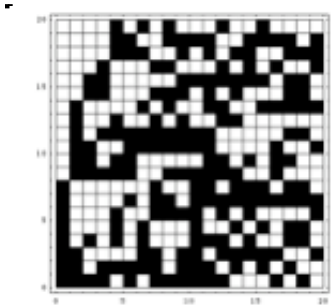


Deletion

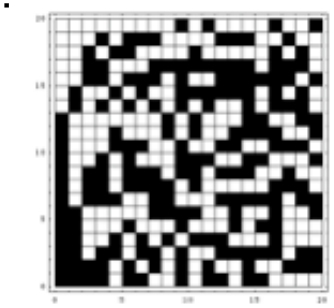


Duplication

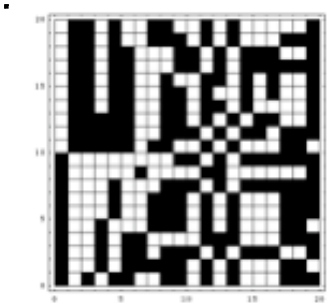
Comparison of Genetic Operators (Genotypes)



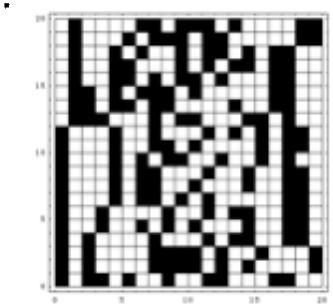
Initial Population



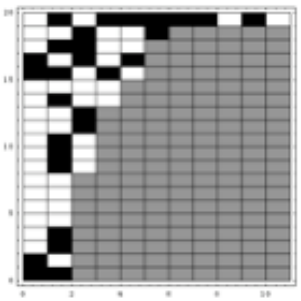
Mutation



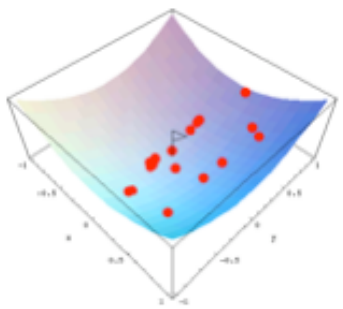
Recombination



Inversion

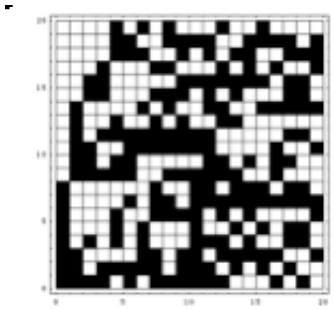


Deletion

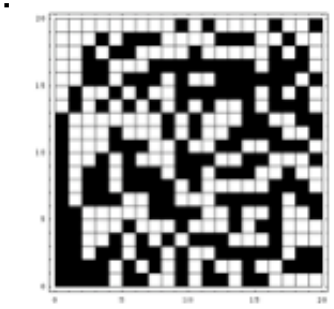


Duplication

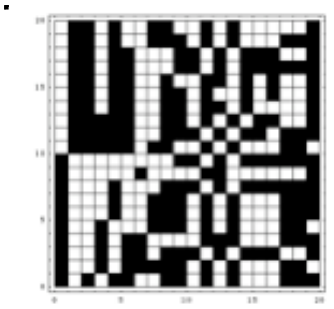
Comparison of Genetic Operators (Genotypes)



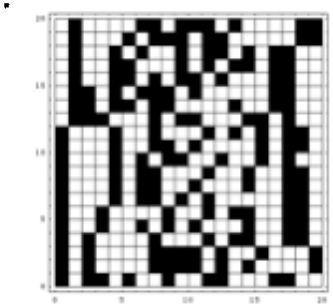
Initial Population



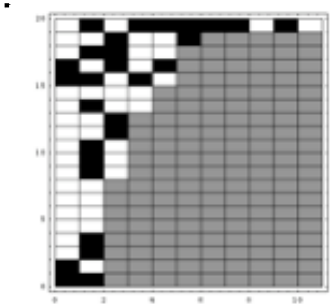
Mutation



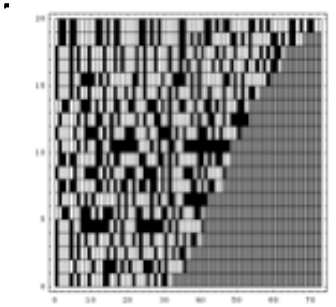
Recombination



Inversion

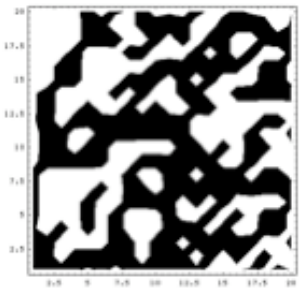


Deletion

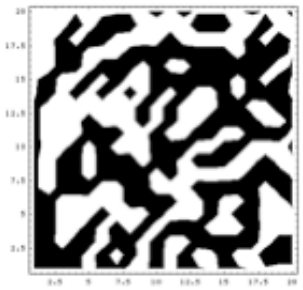


Duplication

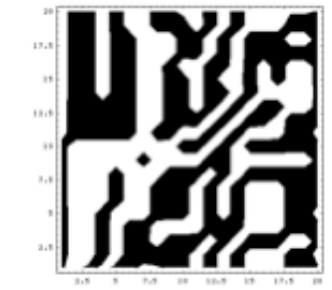
Comparison of Genetic Operators (Genotypes)



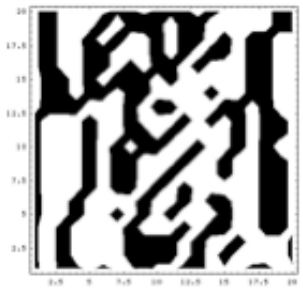
Initial Population



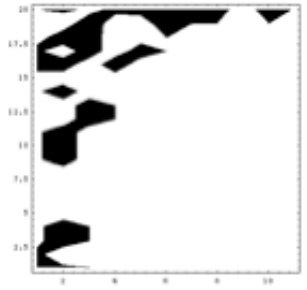
Mutation



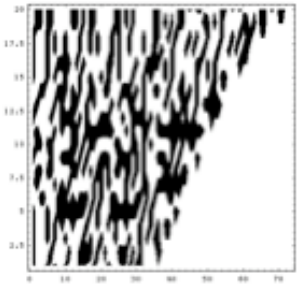
Recombination



Inversion

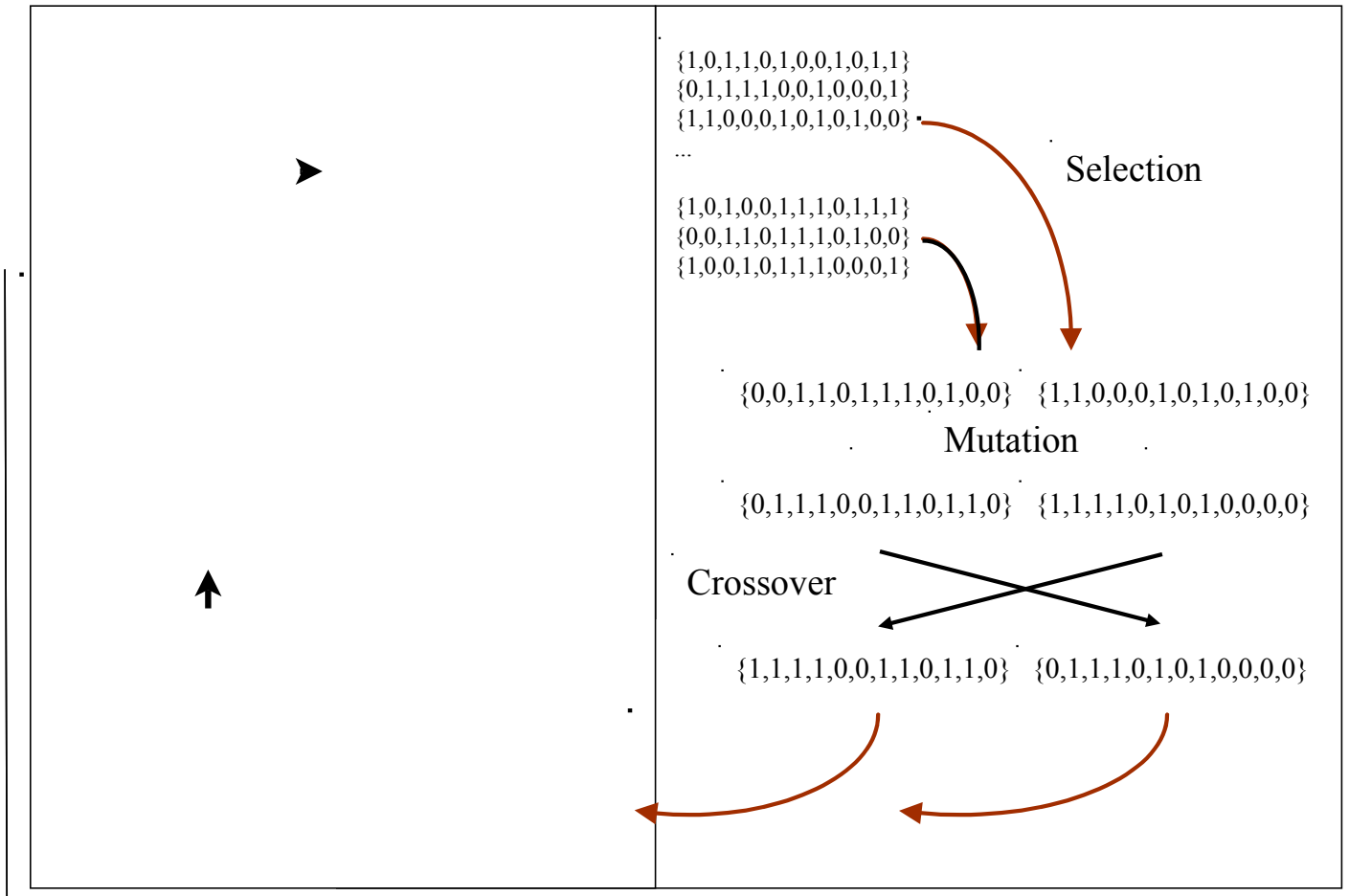


Deletion



Duplication

Phenotypic vs. Genotypic Structures



GA Dualism — Dualism in Nature

Genotype

Phenotype

{1,0,1,1,0,1,0,0,1,0,1,1}
{0,1,1,1,1,0,0,1,0,0,0,1}
{0,0,1,1,0,101,1,0,1,0,0}

...

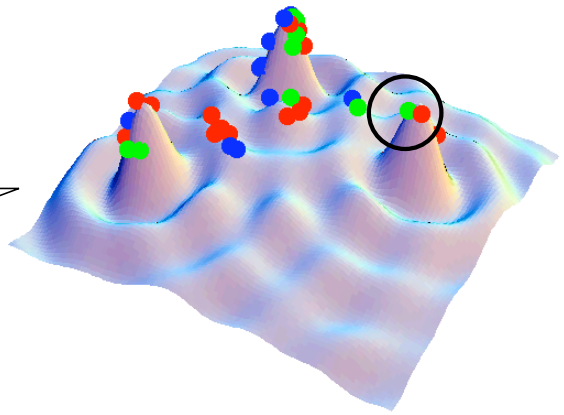
{1,1,0,0,0,1,0,1,0,1,0,0}

...

{1,0,1,0,0,1,1,1,0,1,1,1}
{0,0,1,1,0,1,1,1,0,1,0,0}
{1,0,0,1,0,1,1,1,0,0,0,1}

Decoding

Interpretation



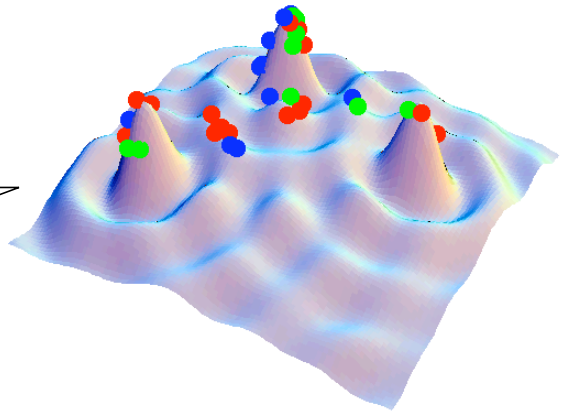
GA Dualism — Dualism in Nature

Genotype

Phenotype

Decoding

Interpretation



GA Dualism — Dualism in Nature

Genotype

Transcription
Translation

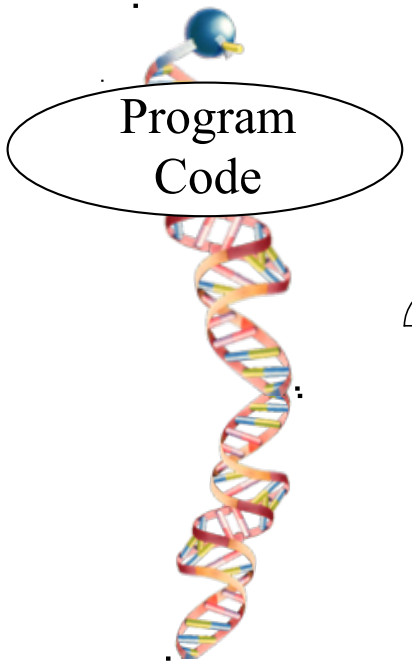
Development
Morphogenesis

Phenotype

GA Dualism — Dualism in Nature

Genotype

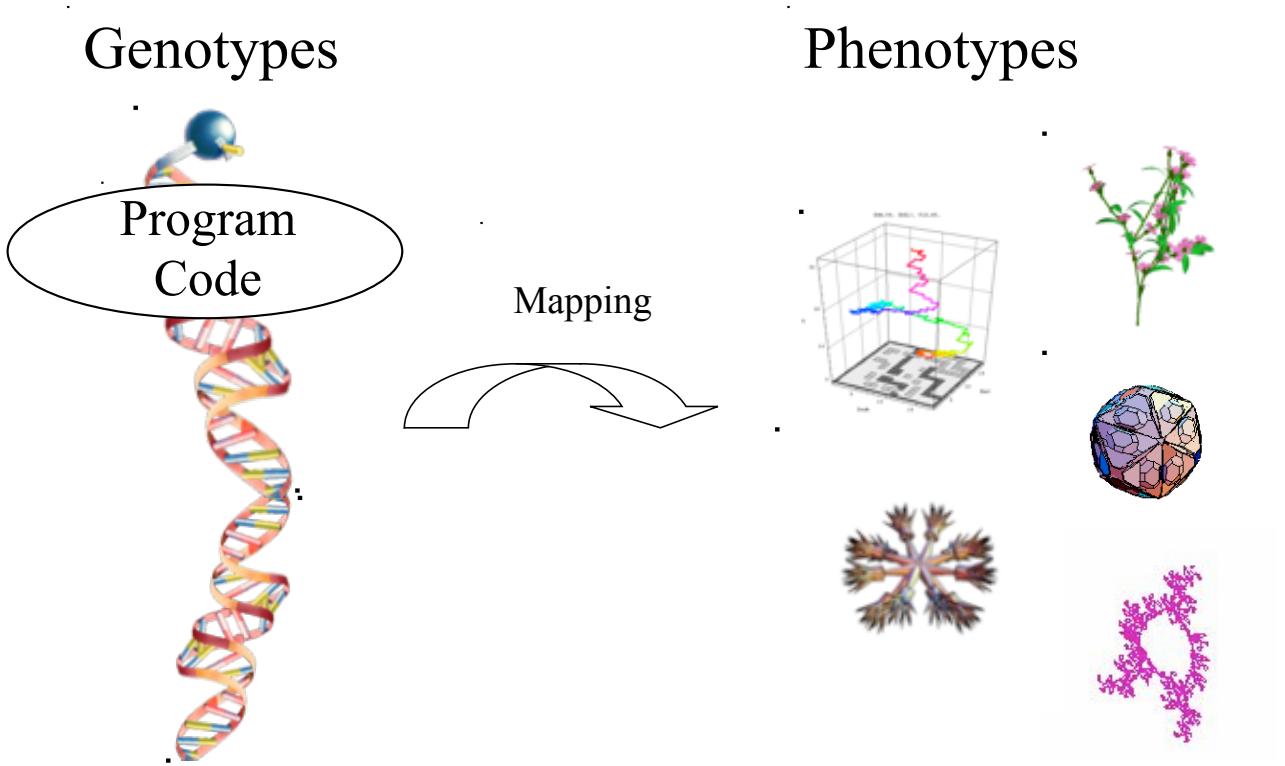
Phenotype



Transcription
Translation

Development
Morphogenesis

GA Dualism — Dualism in Nature



Genetic Programming (GP)

John Koza
(1992)

Programming of Computers
by Means of Natural Selection

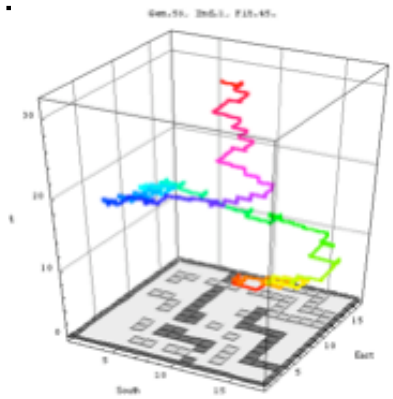
Genetic Computer Programming

or

How to Program a Computer
without
Explicitly Telling It What to Do

Ant Tracker

Breeding
of Simple
Control Programs



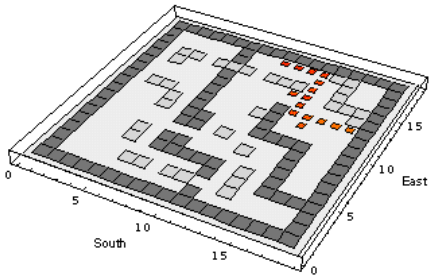
A Classic GP Example

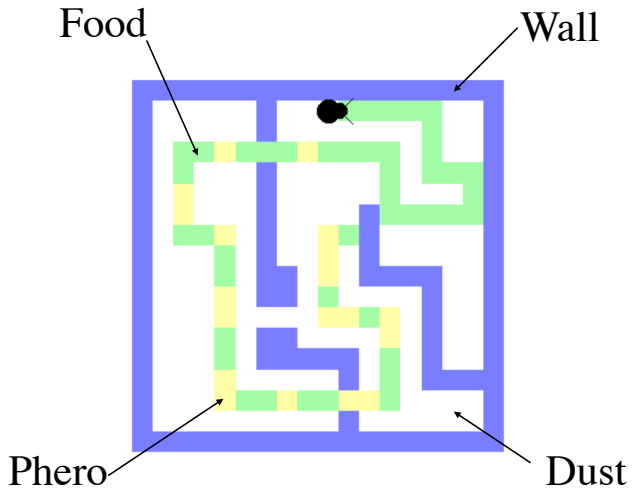
AntTracker: Collecting Food in a Maze

Command building blocks:

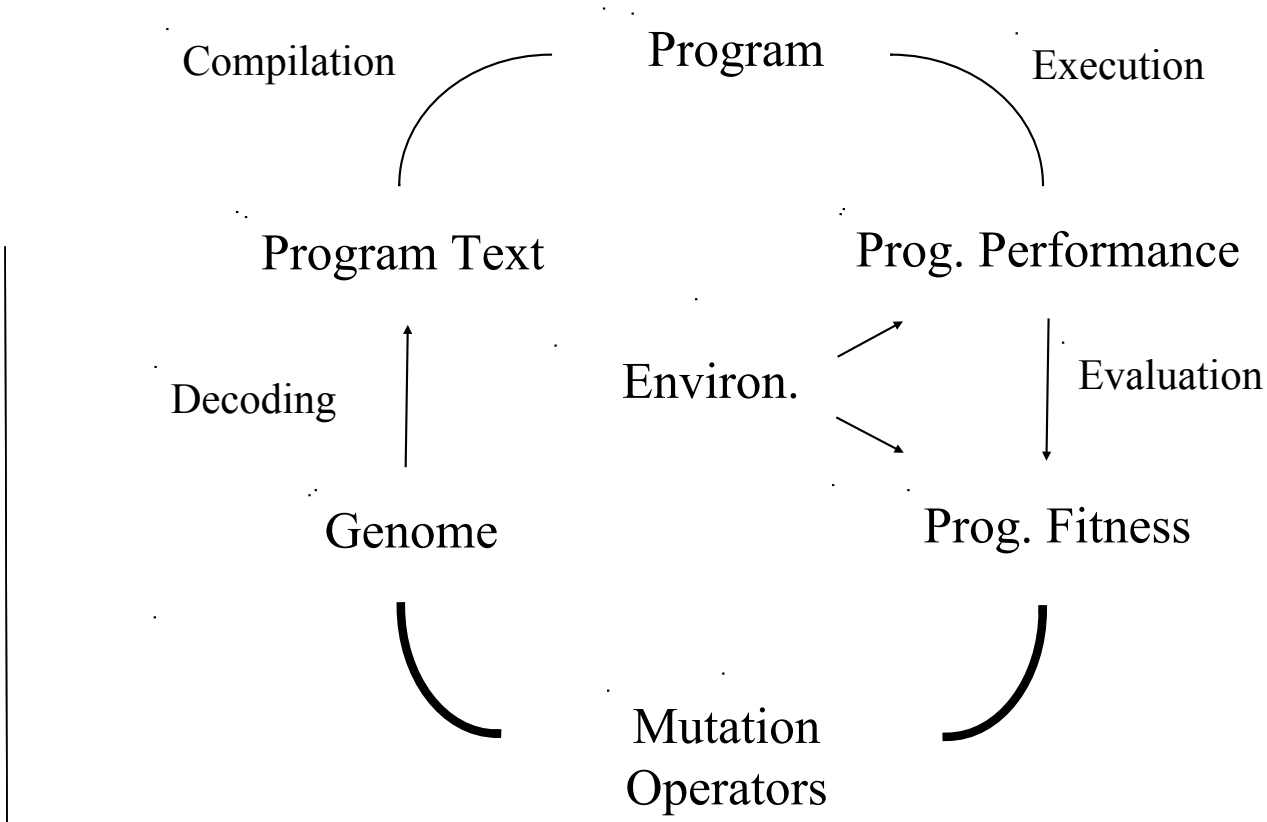
- advance
- turnLeft
- turnRight
- nop

- ifSensor[*food*][**A**, **B**]
  ifSensor[*phero*][**A**, **B**]
  ifSensor[*dust*][**A**, **B**]
  ifSensor[*wall*][**A**, **B**]



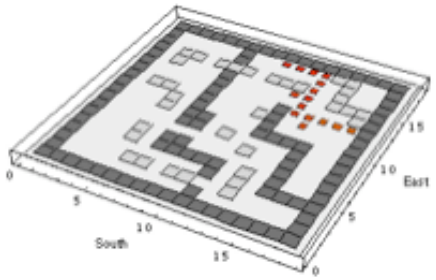


How Genetic Programming Works



AntTracker Evolution

- Fitness criterion:
  Number of collected food pieces
- 50 individuals per population
- 100 generations (max.)
- < 400 action steps
- GP operators:
  - Mutation
  - Crossover
  - Permutation
  - Deletion
  - Duplication
  - En-/decapsulation

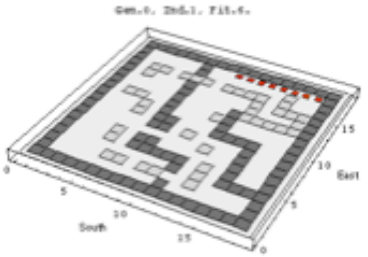


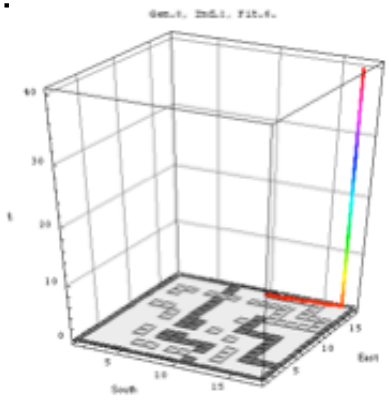
AntTracker: Generation 0

06

Fitness

```
AntTracker[
 seq[advance]
]
```



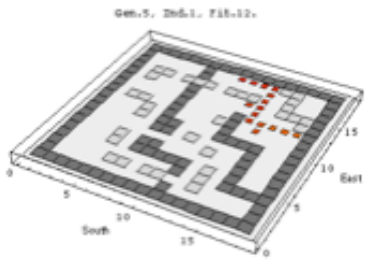


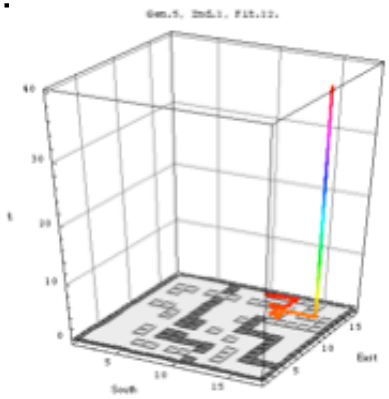
AntTracker: Generation 5

12

Fitness

```
AntTracker[
 seq[seq[advance],
  ifSensor[dust][
   seq[seq[stop]],
   seq[advance]],
  seq[
  ifSensor[dust][
   seq[turnLeft, again],
   seq[nop]],
  seq[seq[advance],
   ifSensor[wall][
    seq[seq[stop],
  seq[nop],
     ifSensor[phero][
      seq[turnLeft],
       seq[stop]]],
    seq[turnRight]]],
  seq[nop]]]]
```



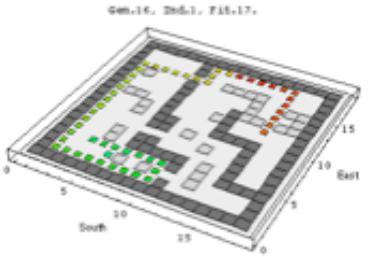


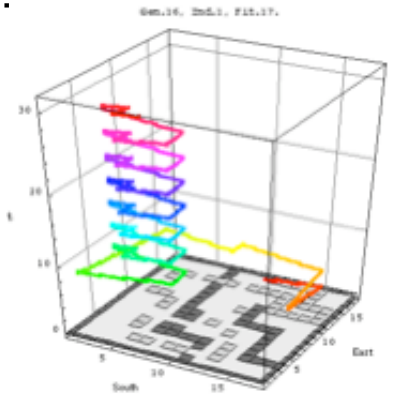
AntTracker: Generation 16

17

Fitness

```
AntTracker[
 seq[seq[advance],
  ifSensor[wall][
   seq[seq[stop], seq[nop],
    ifSensor[phero][
     seq[turnLeft],
     seq[advance]]],
   seq[turnRight]],
  seq[ifSensor[dust][
   seq[turnLeft, again],
    seq[nop]],
   seq[ifSensor[wall][
    seq[seq[turnLeft, again],
     seq[nop],
      ifSensor[phero]
 [seq[stop]]],
  seq[seq[advance]]]],
 seq[nop]]]]
```



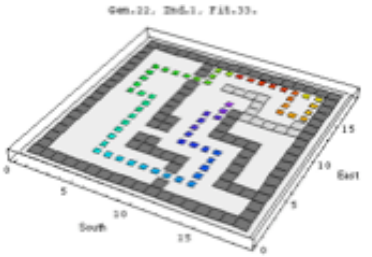


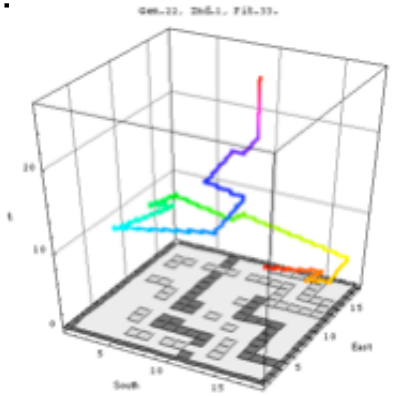
AntTracker: Generation 22

33

Fitness

```
AntTracker[
 seq[
  ifSensor[wall][
   seq[turnLeft, again],
seq[turnRight]],
  seq[
   ifSensor[dust][seq[turnLeft,
    again],seq[nop]],
   seq[
    ifSensor[wall][
     seq[seq[turnLeft, again],
seq[nop],
      ifSensor[phero][seq[stop]]],
     seq[seq[advance]]]],
   seq[nop]]]]
```



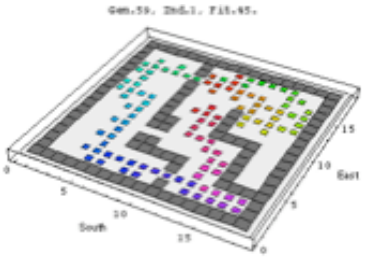


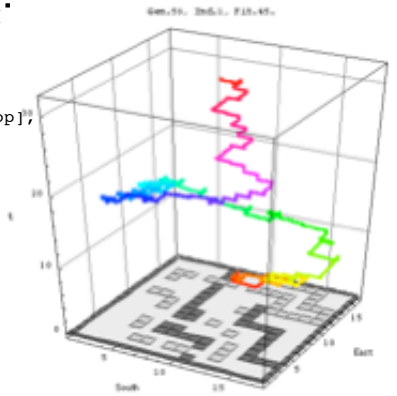
AntTracker: Generation 59

45

Fitness

```
AntTracker[
 seq[
  ifSensor[wall][
   seq[turnLeft, turnLeft, again],
   seq[
    ifSensor[wall][seq[turnLeft, again]
     seq[turnRight]],
    seq[
     seq[
      ifSensor[wall][
       seq[seq[turnLeft, again], seq[no
        ifSensor[wall][seq[again],
seq[seq[nop], seq[nop]]]],
        seq[seq[advance]]]]]],
   seq[
    ifSensor[dust][seq[turnLeft, again]],
    seq[
     ifSensor[wall][seq[ifSensor[dust][
      seq[advance]]],
      seq[ifSensor[dust][seq[turnLeft,
again]],
       seq[ifSensor[wall][
        seq[seq[turnLeft, again], seq[nop],
         ifSensor[dust][seq[stop],
    seq[stop], seq[turnLeft]]],
       seq[seq[advance]]]],
     seq[nop]]]]]]]
```





Gen. 1

```
AntTracke
 seq[adva
]
```

Gen. 5

```
AntTracker[
 seq[seq[advance],
  ifSensor[dust][seq[seq[stop]], seq[advance]],
  seq[
   ifSensor ust][seq[turnLeft, again], seq[nop]],
            ance],
             l][
             op], seq[nop],
            (phero][seq[turnLeft], seq[stop]]
     seq[tu  Right]]],
   seq[nop],
]]
```

Gen. 16

```
AntTracker[
 seq[seq[advance],
  ifSensor[wall][
   seq eq[stop], seq[nop],
    if   sor[phero][seq[turnLeft], seq[advance]]],
         Right]],
          sor[dust][seq[turnLeft, again],seq[nop]],
         ensor[wall][
   se  eq[turnLeft, again], seq[nop],
    i  sensor[phero][seq[stop]]],
  seq[seq[advance]]]],
 seq[nop]]]]
```

Gen. 59

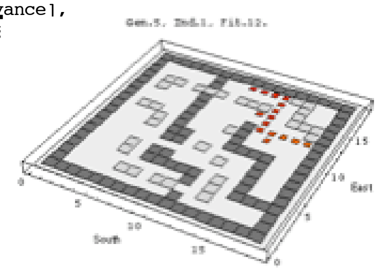
```
AntTracker[
 seq[
  ifSensor[wall][
   seq[turnLeft, turnLeft, again],
   seq[
    ifSensor[wall][seq[turnLeft, again], seq[turnRight]],
    seq[
     seq[
      ifSensor[wall][
       seq[seq[turnLeft, again], seq[nop],
        ifSensor[wall][seq[again], seq[seq[nop   seq[nop]]]]
       seq[seq[advance]]]]]]],
    seq[
     ifSensor[dust][seq[turnLeft, again]],
     seq[
      ifSensor[wall][seq[ifSensor[dust][seq[adv   e]]],
      seq[ifSensor[dust][seq[turnLeft, again]],
       seq[ifSensor[wall][
        seq[seq[turnLeft, again], seq[nop],
         ifSensor[dust][seq[stop], seq[stop], seq[turnLeft]]],
        seq[seq[advance]]]],
      seq[nop]]]]]]]
```

Gen.

```
AntTracker[
 seq[
  ifSensor[wall][
   seq[turnLeft, again], seq[turnRight]],
   seq[
    ifSensor[dust][seq[turnLeft, again],seq[nop]],
    seq[
     ifSensor[wall][
      seq[seq[turnLeft, again], seq[nop],
       ifSensor[phero][seq[stop]]],
      seq[seq[advance]]]],
    seq[nop]]
]]
```

Gen. 5

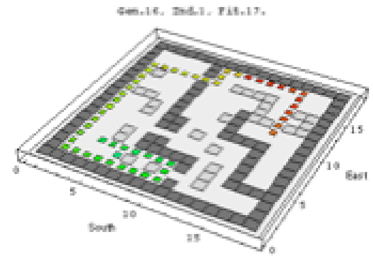
```
AntTracker[
 seq[seq[advance],
  ifSensor[c
  seq[
   ifSensor
```



```
    seq[tu
   seq[nop],
]]
```

Gen. 16

```
AntTracker[
 seq[seq[advan
  ifSensor[wal
   seq  eq[sto
          sor[p
          Rig
           sor
          enso
   se   eq[tu
    i  Sensor[
   seq[seq[adv
  seq[nop]]]]
```

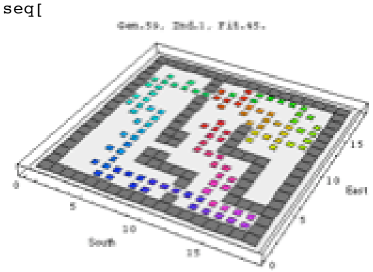


```
AntTracke
 seq[adva
]
```

Gen. 1

```
AntTracker[
 seq[
  ifSensor[wall][
   seq[turnLeft, turnLeft, again],
   seq[
    ifSensor[wall][seq[turnLeft, again], seq[turnRight]],
    seq[
```

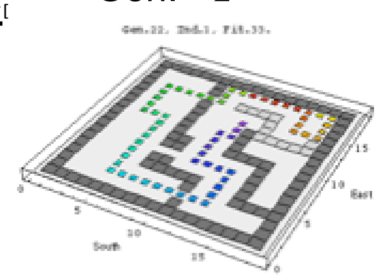
Gen. 59



```
                 ],
                 eq[nop    seq[nop]]]],
                 ,
                 eq[adv    e]]],
                 ain]],
                 ],
     ifSensor[dust][seq[stop], seq[stop], seq[turnLeft]]],
    seq[seq[advance]]]],
   seq[nop]]]]]]
```

Gen.

```
AntTracker[
 seq[
  ifSensor
   seq[tur
  seq[
   ifSenso
   seq[
    ifSens
     seq[s
      ifSe
     seq[s
   seq[nop
]]
```



Reuse of Learnt Building Blocks (1)

Gen. 1

```
AntTracker[
 seq[advance]
]
```

Gen. 5

```
AntTracker[
 seq[seq[advance],
  ifSensor[dust][seq[seq[stop]], seq[advance]],
  seq[
   ifSensor[dust][seq[turnLeft, again], seq[nop]],
   seq[seq[advance],
    ifSensor[wall][
     seq[seq[stop], seq[nop],
      ifSensor[phero][seq[turnLeft], seq[stop]]],
     seq[turnRight]]],
   seq[nop]]
 ]]
```

Gen. 16

```
AntTracker[
 seq[seq[advance],
  ifSensor[wall][
   seq[seq[stop], seq[nop],
    ifSensor[phero][seq[turnLeft], seq[advance]]],
   seq[turnRight]],
  seq[ifSensor[dust][seq[turnLeft, again],seq[nop]],
   seq[ifSensor[wall][
    seq[seq[turnLeft, again], seq[nop],
     ifSensor[phero][seq[stop]]],
   seq[seq[advance]]]],
  seq[nop]]]]
```

Gen. 59

```
AntTracker[
 seq[
  ifSensor[wall][
   seq[turnLeft, turnLeft, again],
   seq[
    ifSensor[wall][seq[turnLeft, again], seq[turnRight]],
    seq[
     seq[
      ifSensor[wall][
       seq[seq[turnLeft, again], seq[nop],
        ifSensor[wall][seq[again], seq[seq[nop], seq[nop]]]],
       seq[seq[advance]]]]]]],
    seq[
     ifSensor[dust][seq[turnLeft, again]],
     seq[
      ifSensor[wall][seq[ifSensor[dust][seq[advance]]],
      seq[ifSensor[dust][seq[turnLeft, again]],
       seq[ifSensor[wall][
        seq[seq[turnLeft, again], seq[nop],
         ifSensor[dust][seq[stop], seq[stop], seq[turnLeft]]],
        seq[seq[advance]]]],
      seq[nop]]]]]]]
```

Gen. 22

```
AntTracker[
 seq[
  ifSensor[wall][
   seq[turnLeft, again], seq[turnRight]],
   seq[
    ifSensor[dust][seq[turnLeft, again],seq[nop]],
    seq[
     ifSensor[wall][
      seq[seq[turnLeft, again], seq[nop],
       ifSensor[phero][seq[stop]]],
      seq[seq[advance]]]],
    seq[nop]]
 ]]
```

Reuse of Learnt Building Blocks (2)

Gen. 1

```
AntTracker[
 seq[advance]
]
```

Gen. 5

```
AntTracker[
 seq[seq[advance],
  ifSensor[dust][seq[seq[stop]], seq[advance]],
  seq[
   ifSensor[dust][seq[turnLeft, again], seq[nop]],
   seq[seq[advance],
    ifSensor[wall][
     seq[seq[stop], seq[nop],
      ifSensor[phero][seq[turnLeft], seq[stop]]],
     seq[turnRight]]],
   seq[nop]]
 ]]
```

Gen. 16

```
AntTracker[
 seq[seq[advance],
  ifSensor[wall][
   seq[seq[stop], seq[nop],
    ifSensor[phero][seq[turnLeft], seq[advance]]],
   seq[turnRight]],
  seq[ifSensor[dust][seq[turnLeft, again],seq[nop]],
   seq[ifSensor[wall][
    seq[seq[turnLeft, again], seq[nop],
     ifSensor[phero][seq[stop]]],
   seq[seq[advance]]]],
  seq[nop]]]]
```

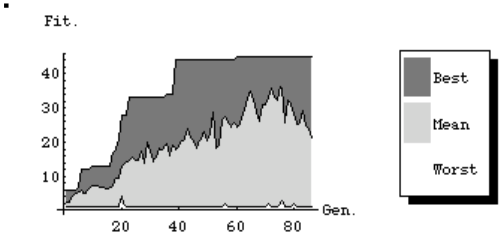
Gen. 59

```
AntTracker[
 seq[
  ifSensor[wall][
   seq[turnLeft, turnLeft, again],
   seq[
    ifSensor[wall][seq[turnLeft, again], seq[turnRight]],
    seq[
     seq[
      ifSensor[wall][
       seq[seq[turnLeft, again], seq[nop],
        ifSensor[wall][seq[again], seq[seq[nop], seq[nop]]],
        seq[seq[advance]]]]]]],
    seq[
     ifSensor[dust][seq[turnLeft, again]],
     seq[
      ifSensor[wall][seq[ifSensor[dust][seq[advance]]],
      seq[ifSensor[dust][seq[turnLeft, again]],
       seq[ifSensor[wall][
        seq[seq[turnLeft, again], seq[nop],
         ifSensor[dust][seq[stop], seq[stop], seq[turnLeft]]],
        seq[seq[advance]]]],
      seq[nop]]]]]]]
```

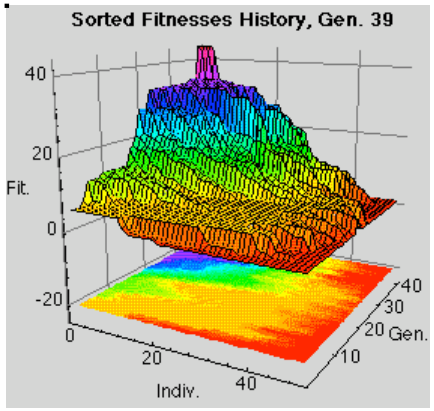
Gen. 22

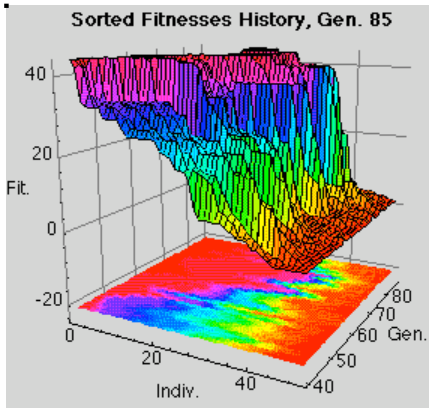
```
AntTracker[
 seq[
  ifSensor[wall][
   seq[turnLeft, again], seq[turnRight]],
   seq[
    ifSensor[dust][seq[turnLeft, again],seq[nop]],
    seq[
     ifSensor[wall][
      seq[seq[turnLeft, again], seq[nop],
       ifSensor[phero][seq[stop]]],
      seq[seq[advance]]]],
    seq[nop]]
 ]]
```

AntTracker: Evolution of Fitnesses



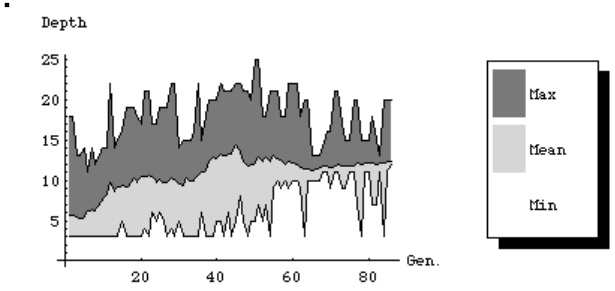
Fitness evolution of *AntTracker* program genomes



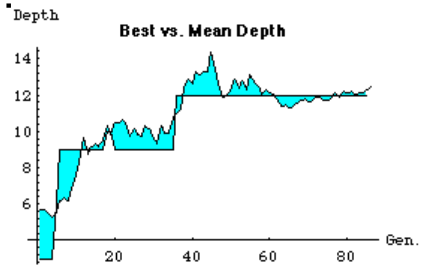


[Jacob 2001, Ch. 9]

AntTracker: Evolution of Program Depth



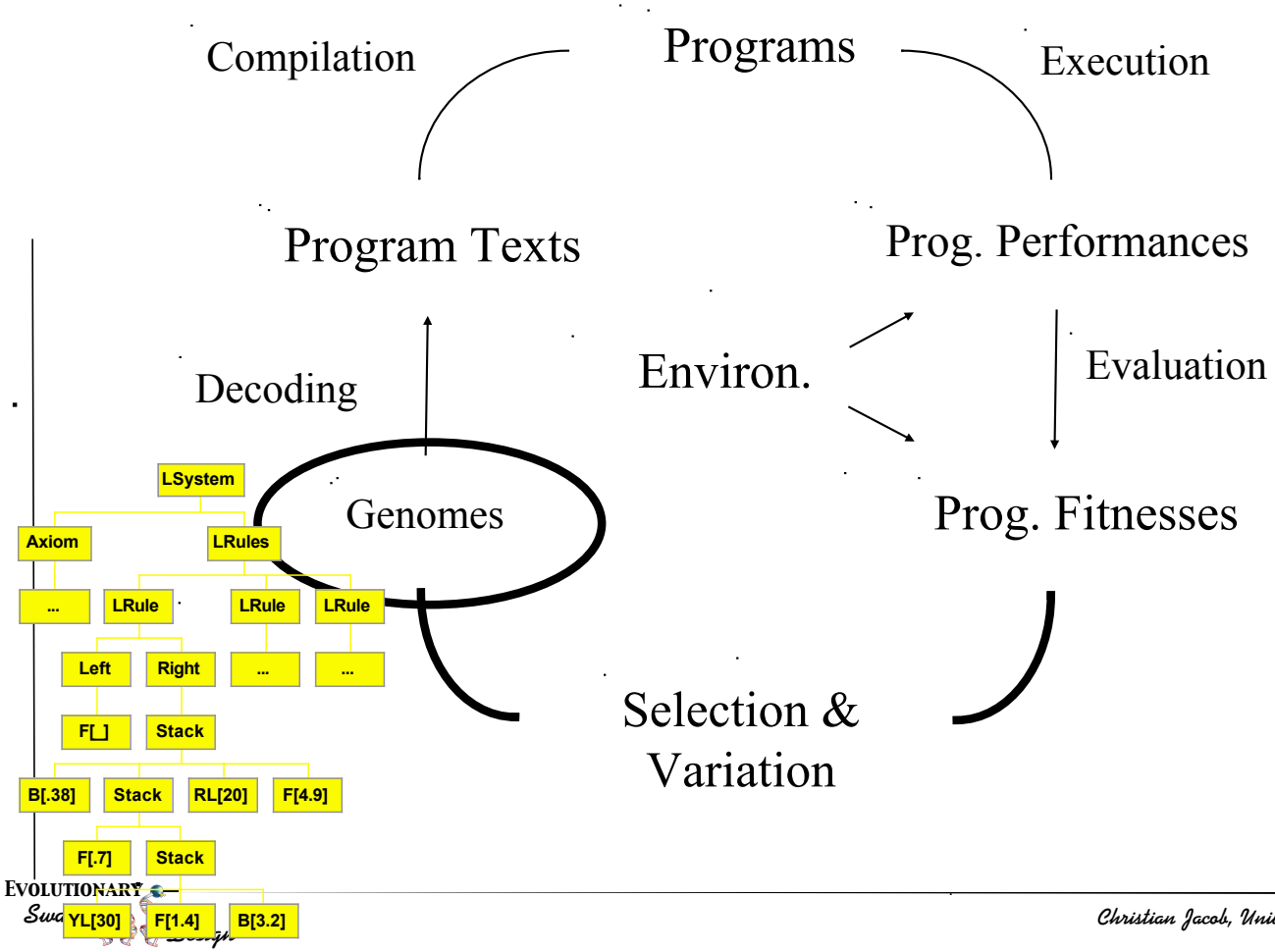
Evolution of program expression depth



Depth of the best program per generation versus average program depth

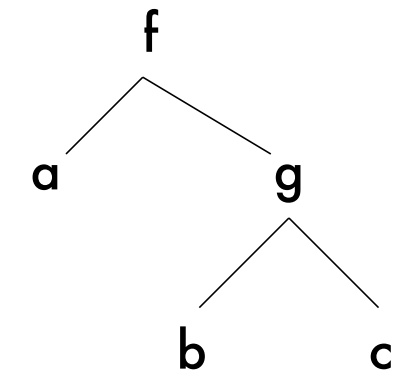
[Jacob 2001, Ch. 9]

How to Generate a 'Gene Pool' of Programs

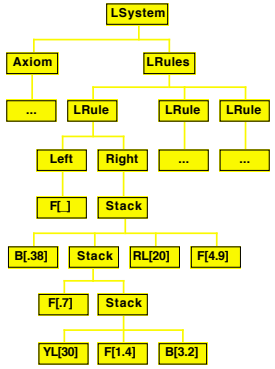


Formulas / Expressions as 'Tree Structures'

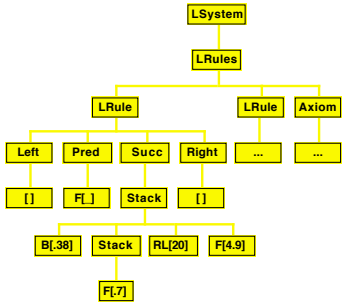
f(a, g(b, c)) ⟺



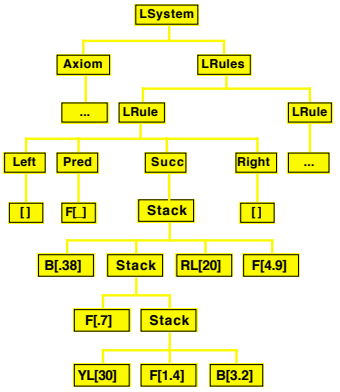
Population of Symbolic Expressions



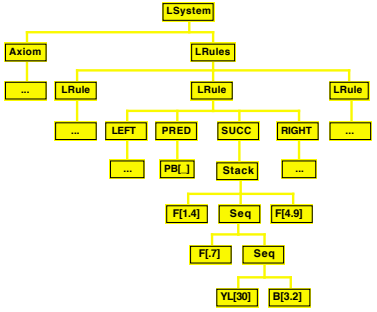
Program 1



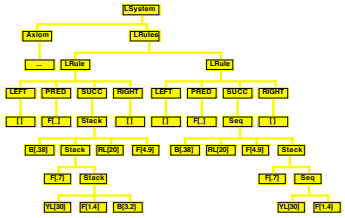
Program 2



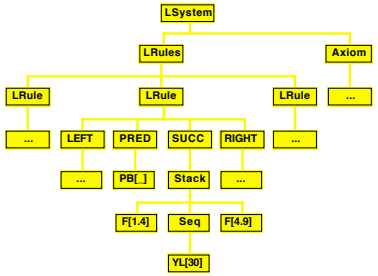
Program 3



Program 4

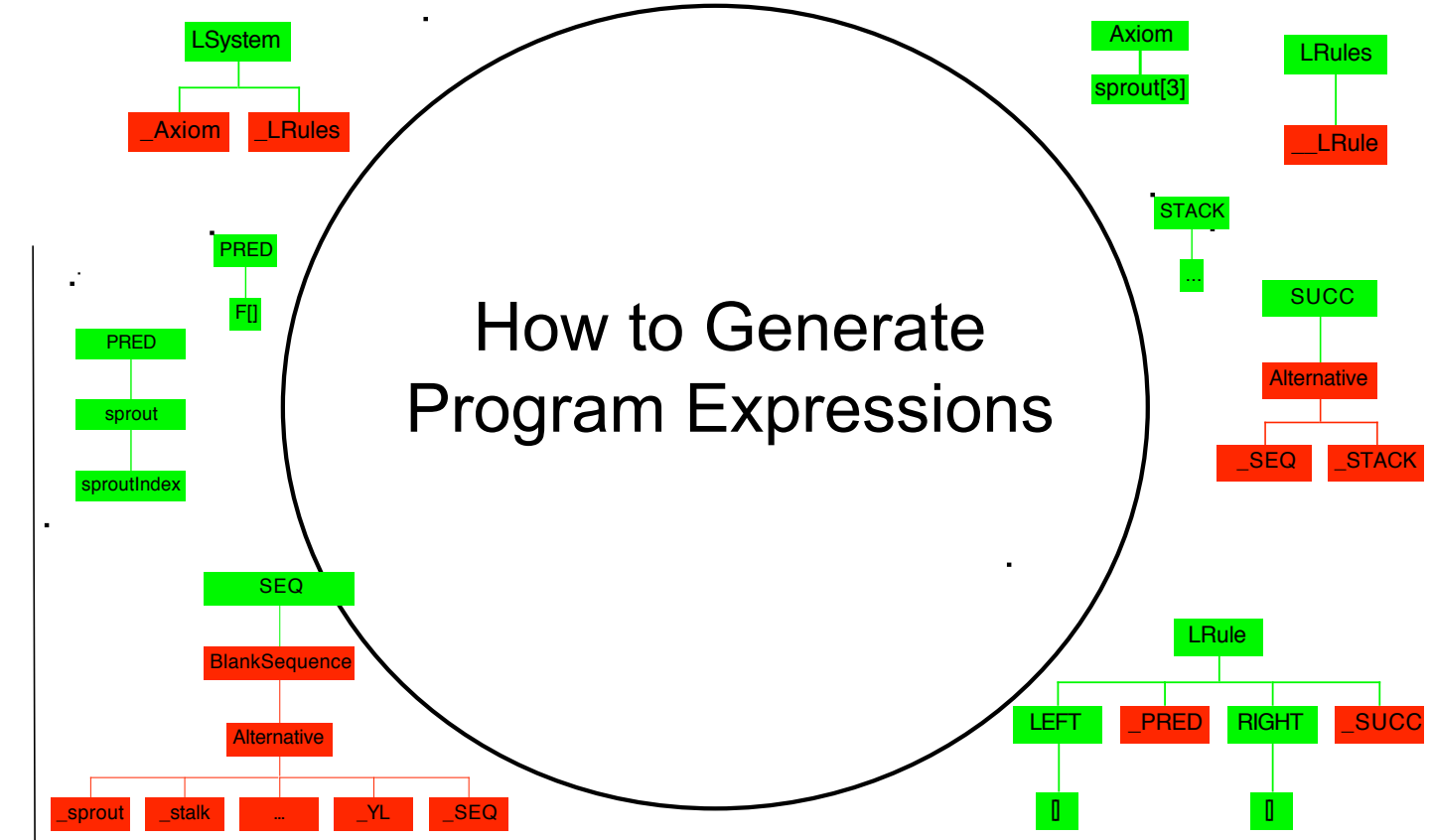


Program 5

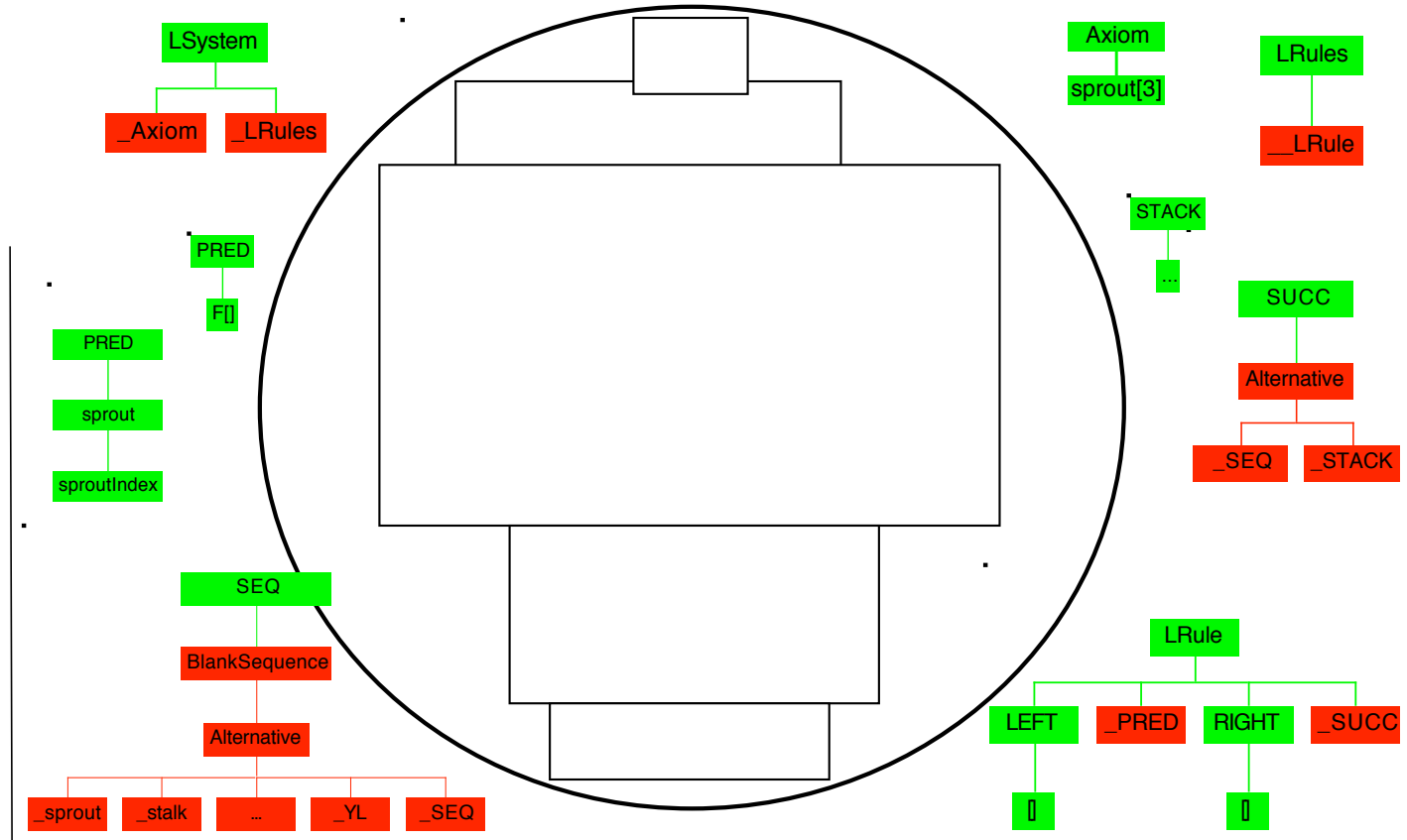


Program 6

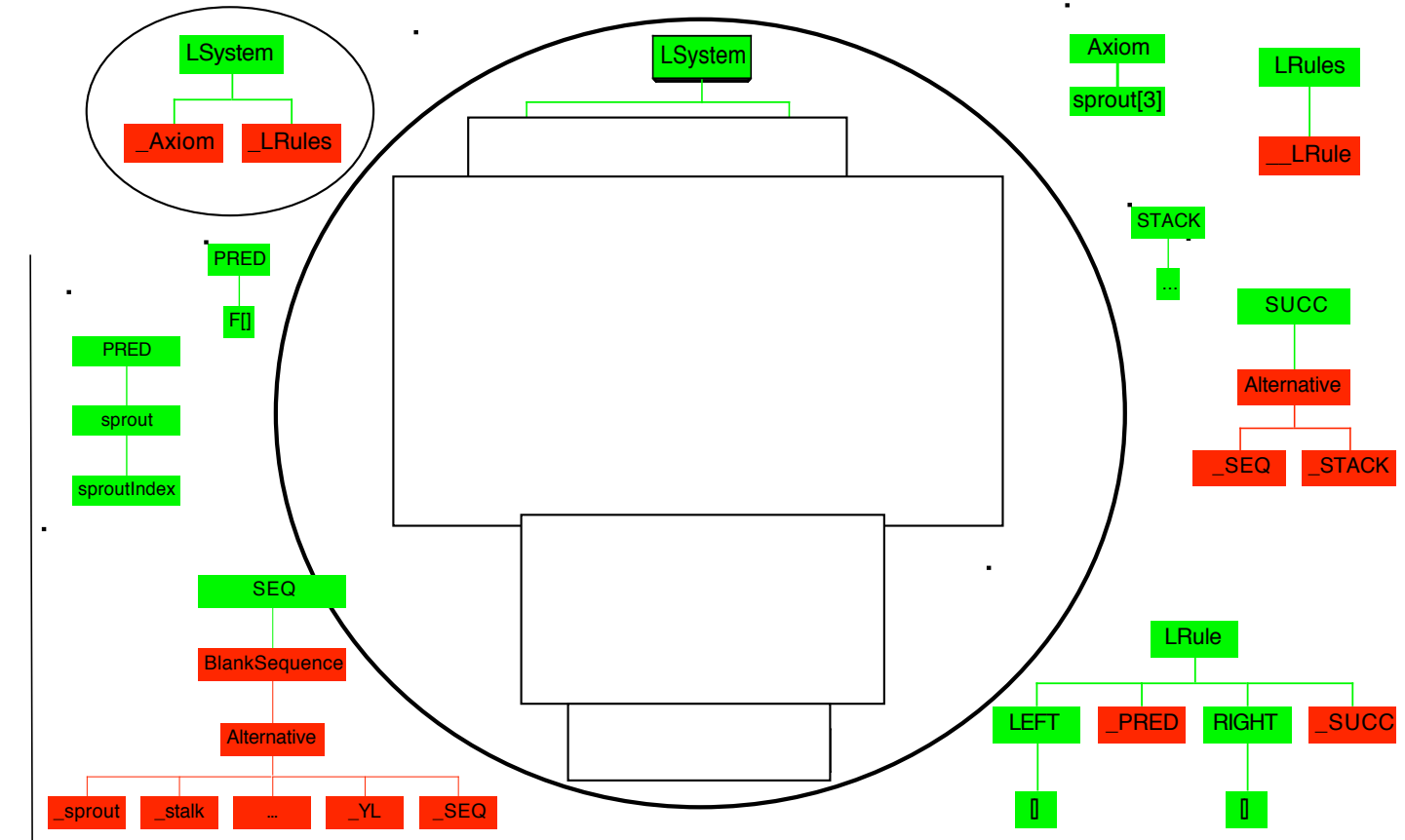
How to Generate Program Expressions



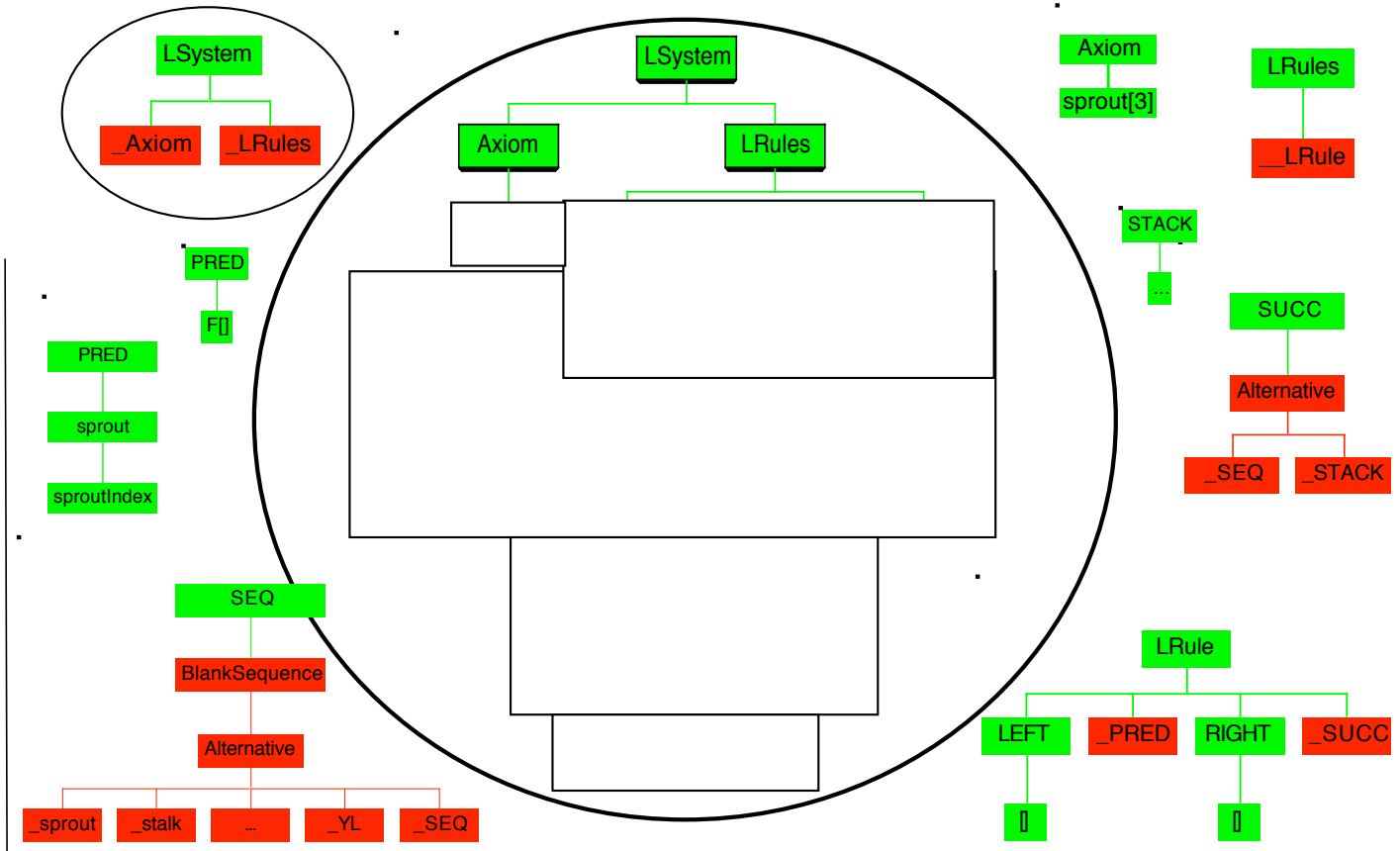
Generating program expressions



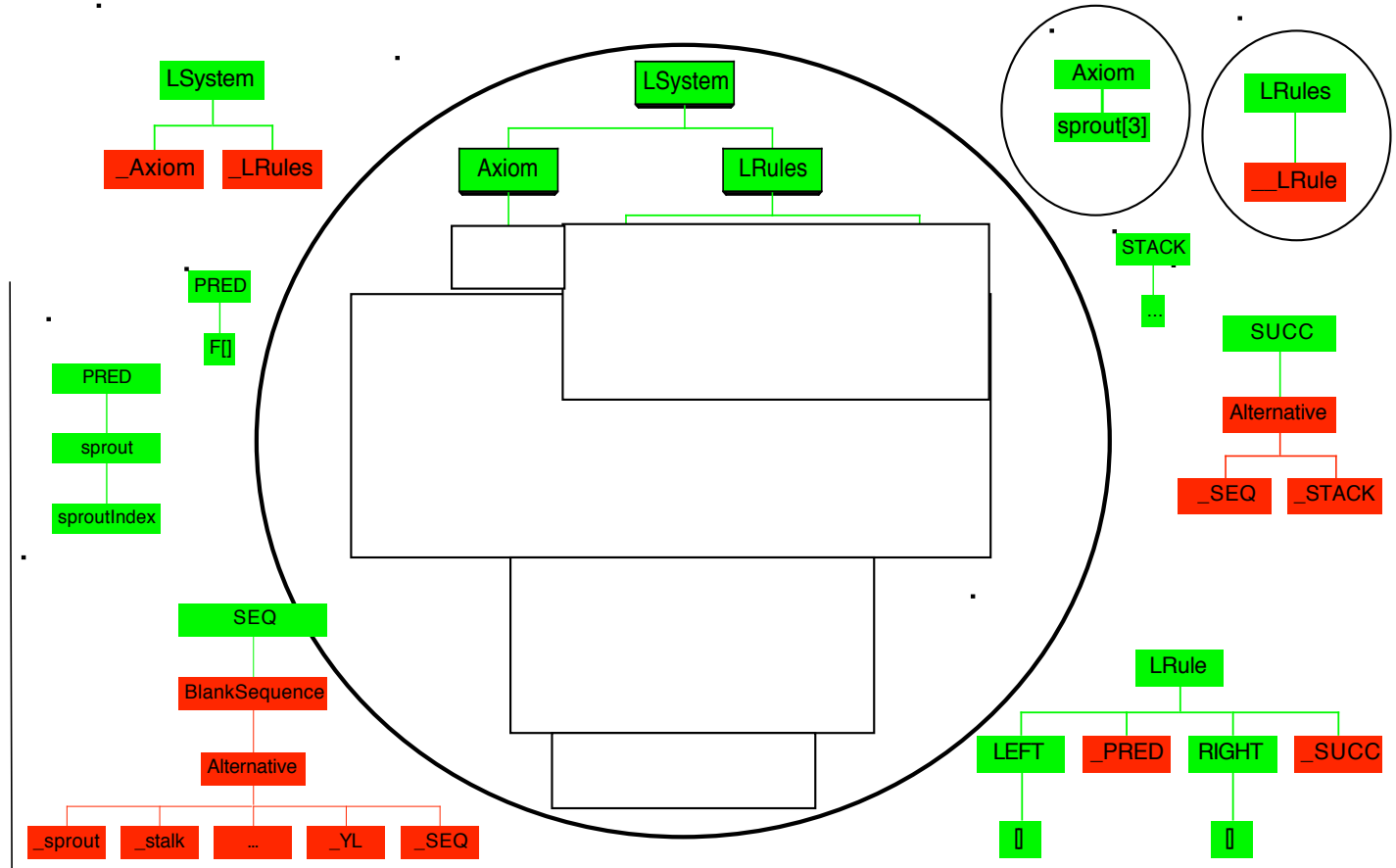
Generating program expressions



Generating program expressions

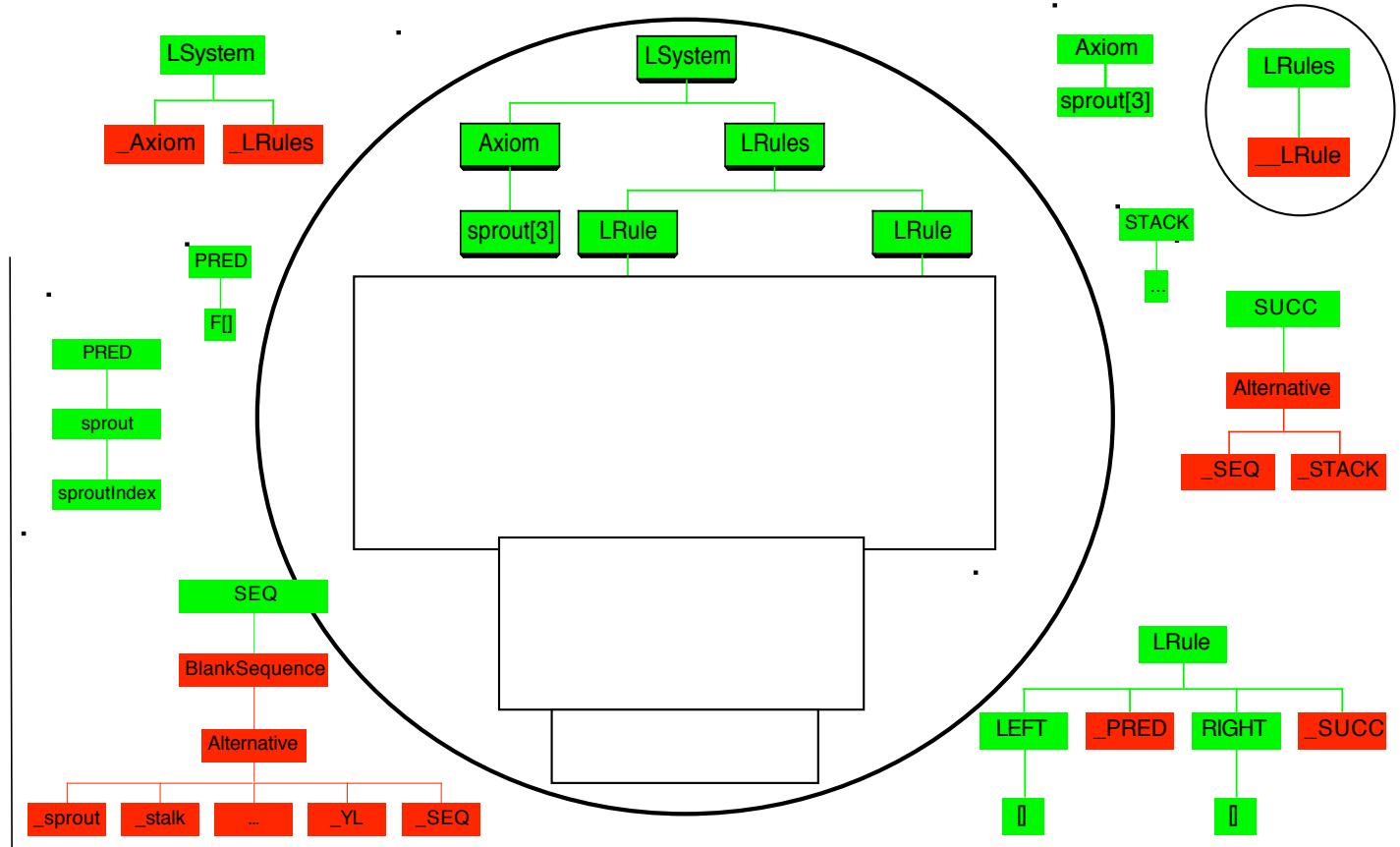


Generating program expressions



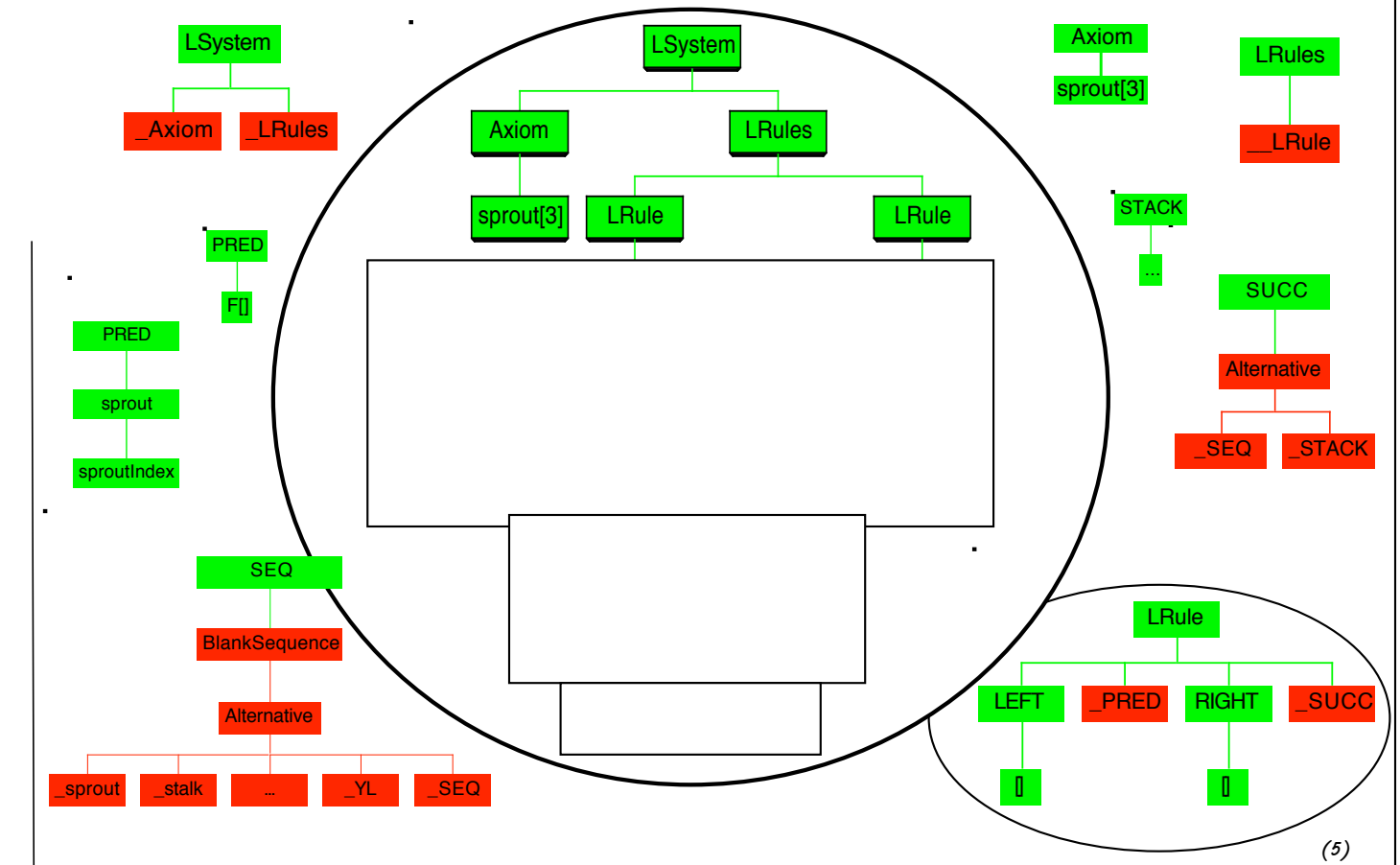
(4)

Generating program expressions



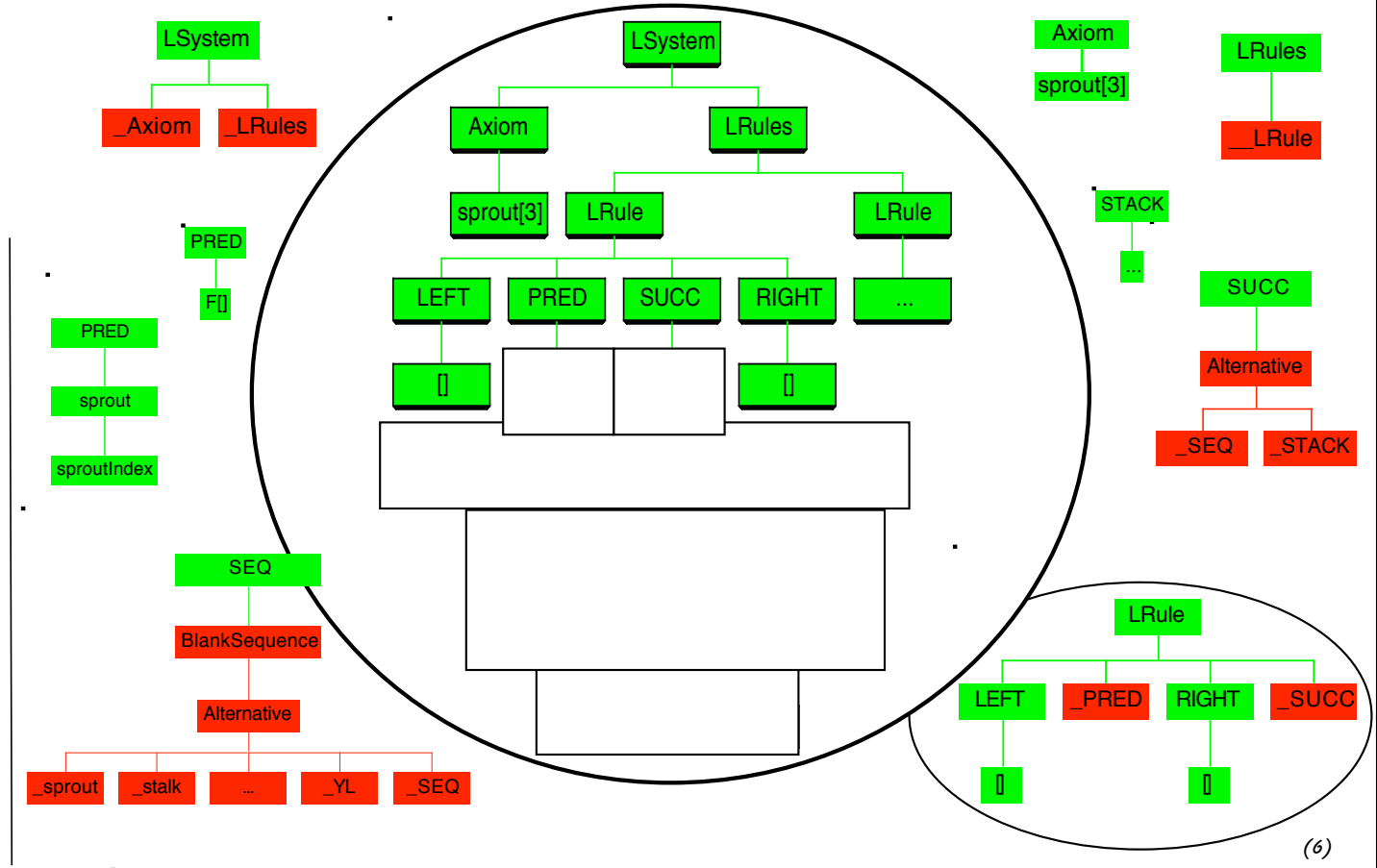
(5)

Generating program expressions



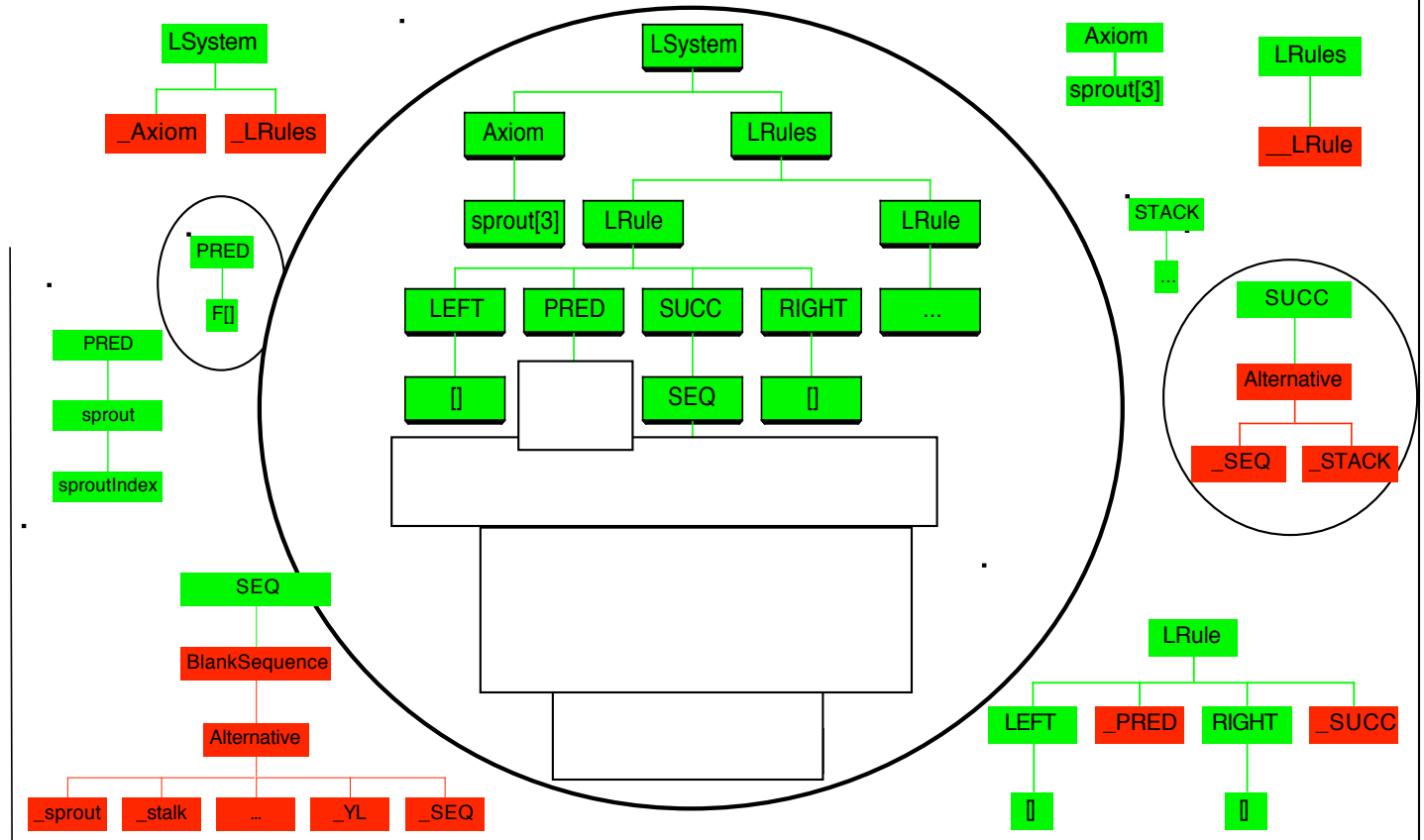
(5)

Generating program expressions

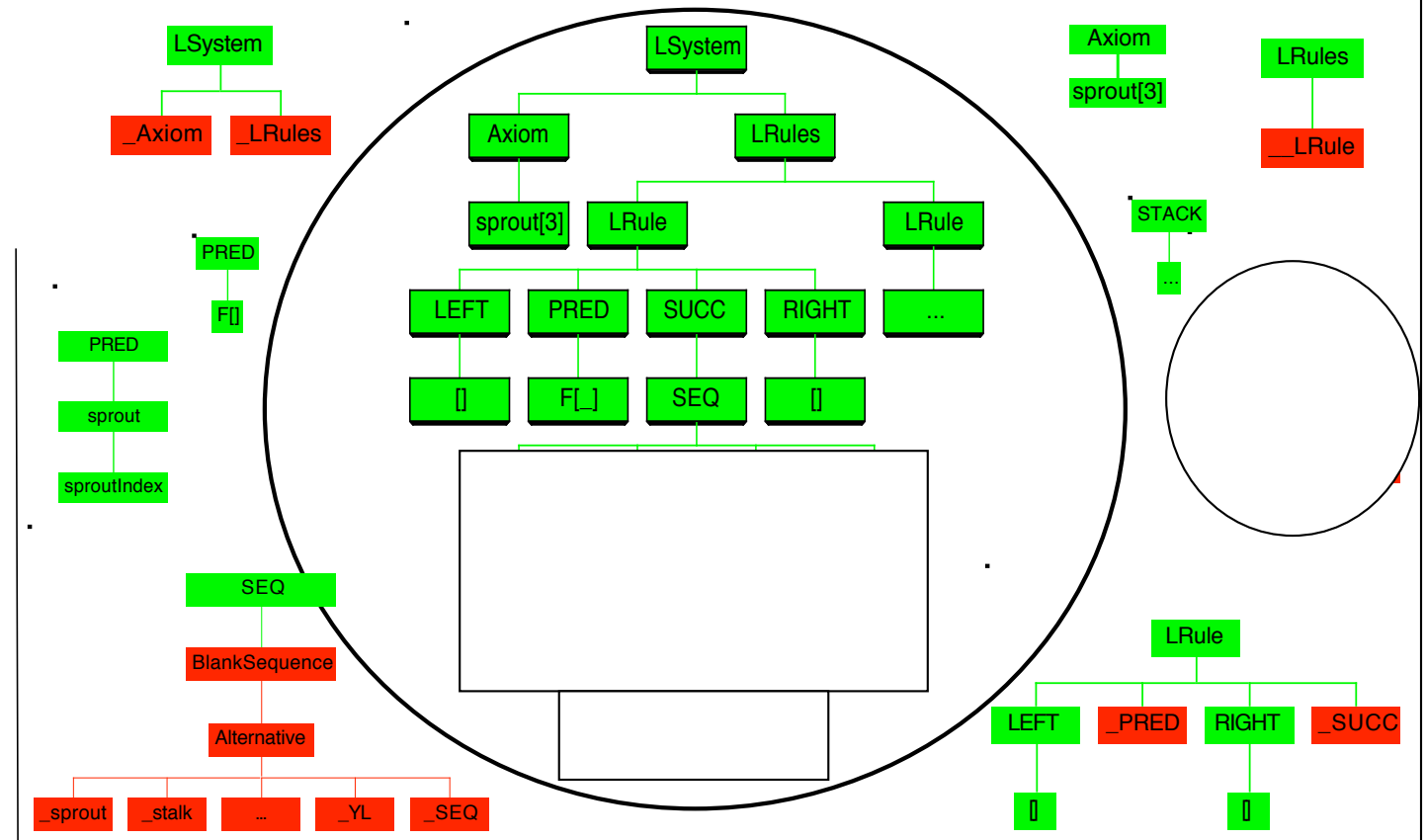


(6)

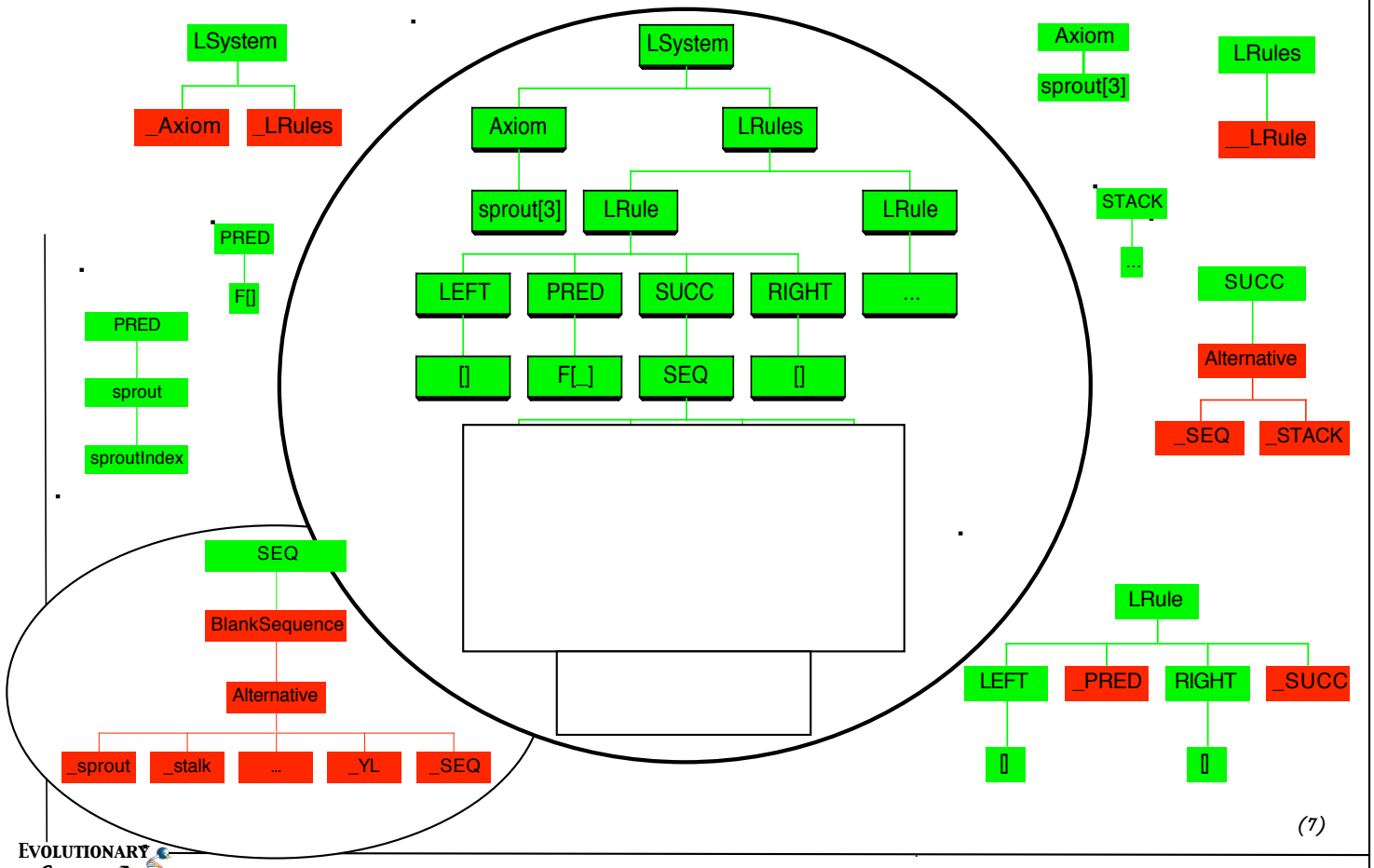
Generating program expressions



Generating program expressions

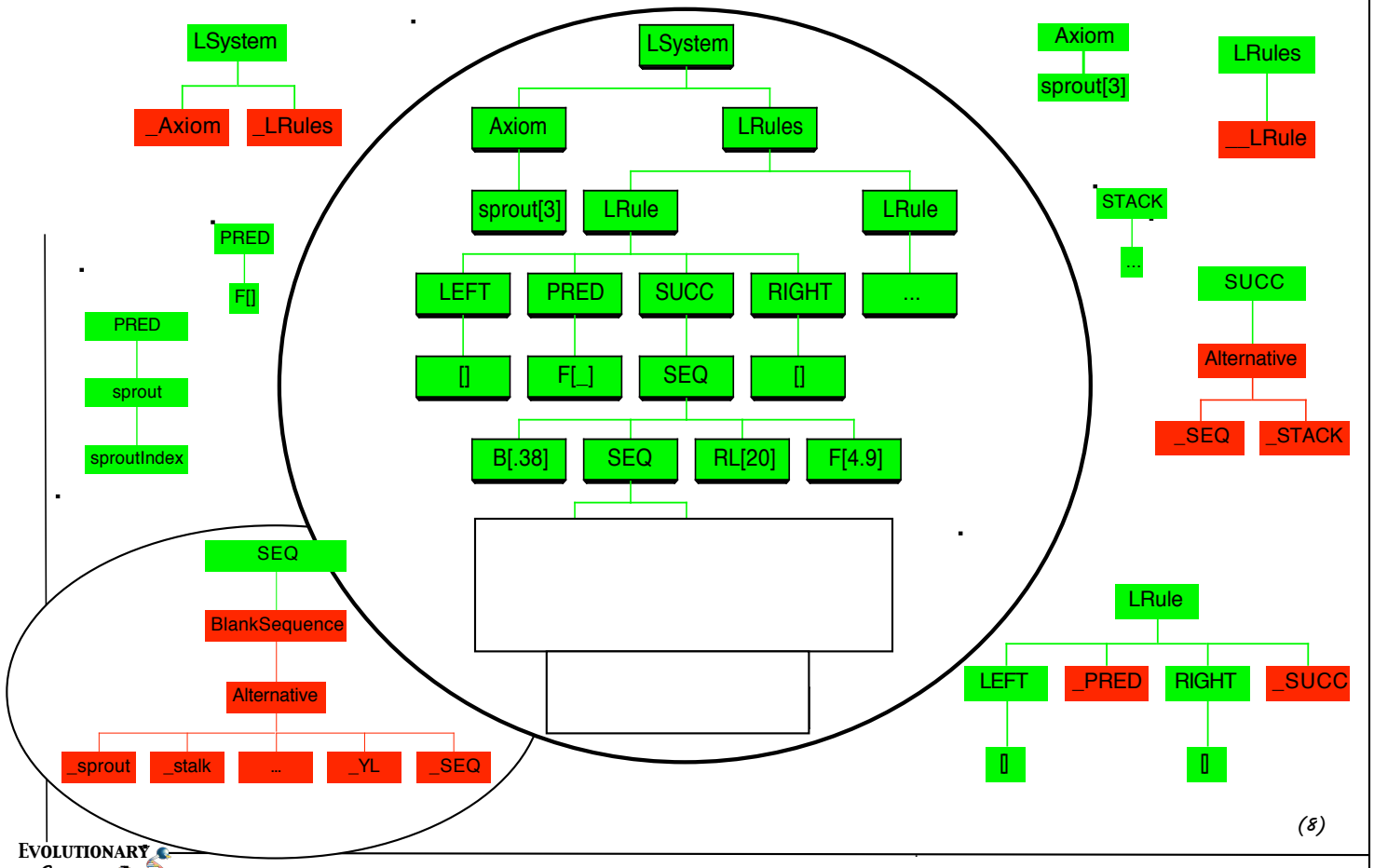


Generating program expressions



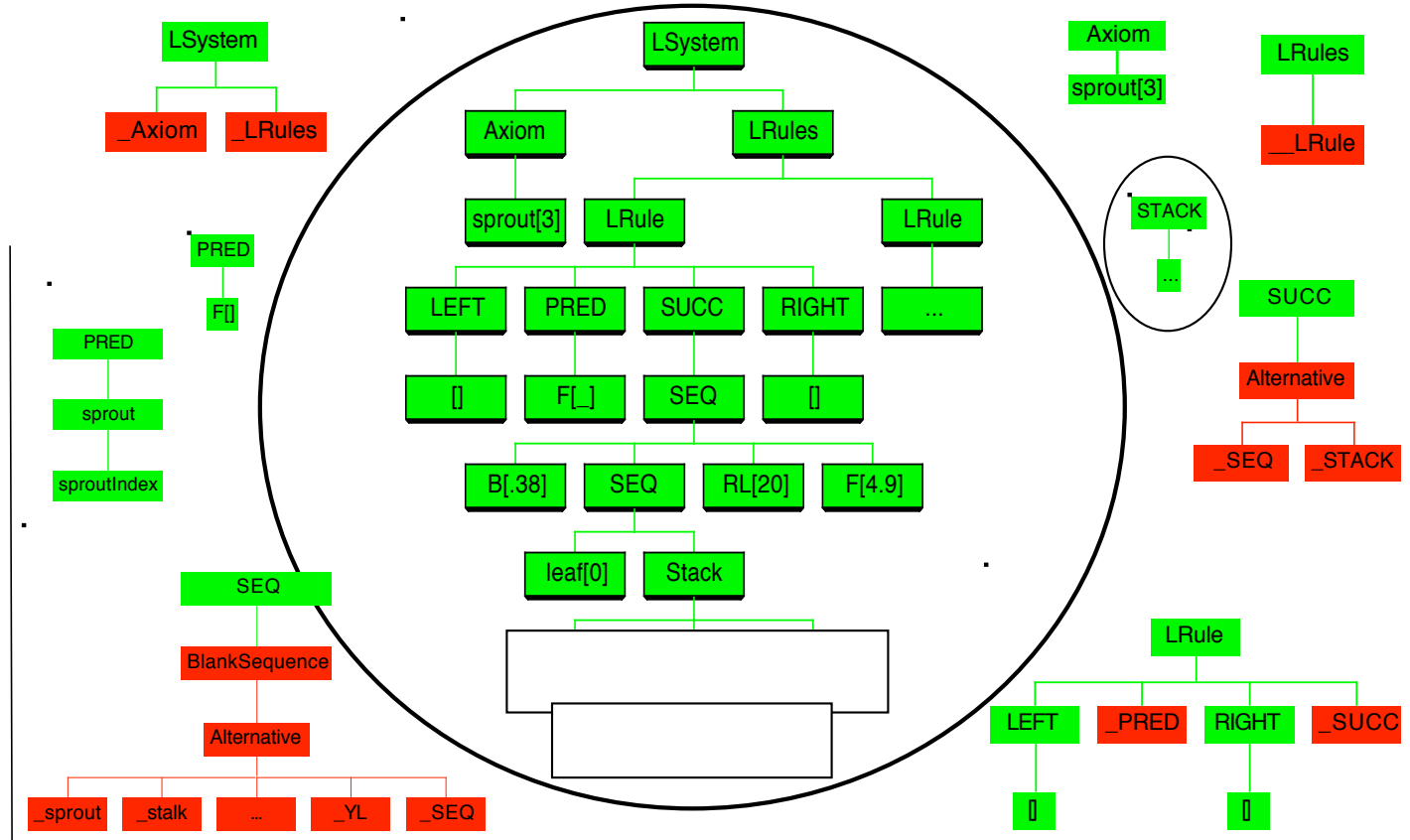
(7)

Generating program expressions



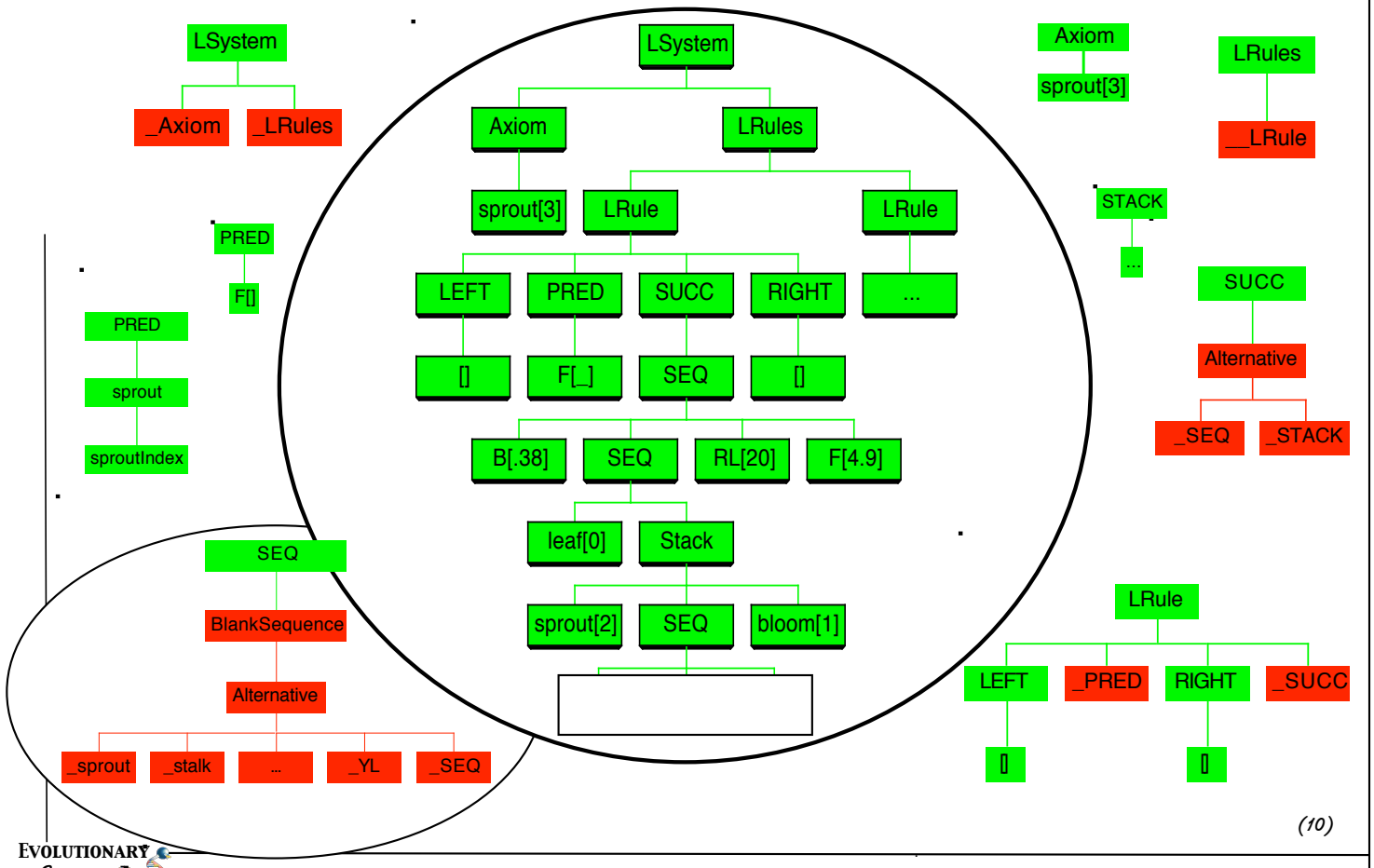
(8)

Generating program expressions



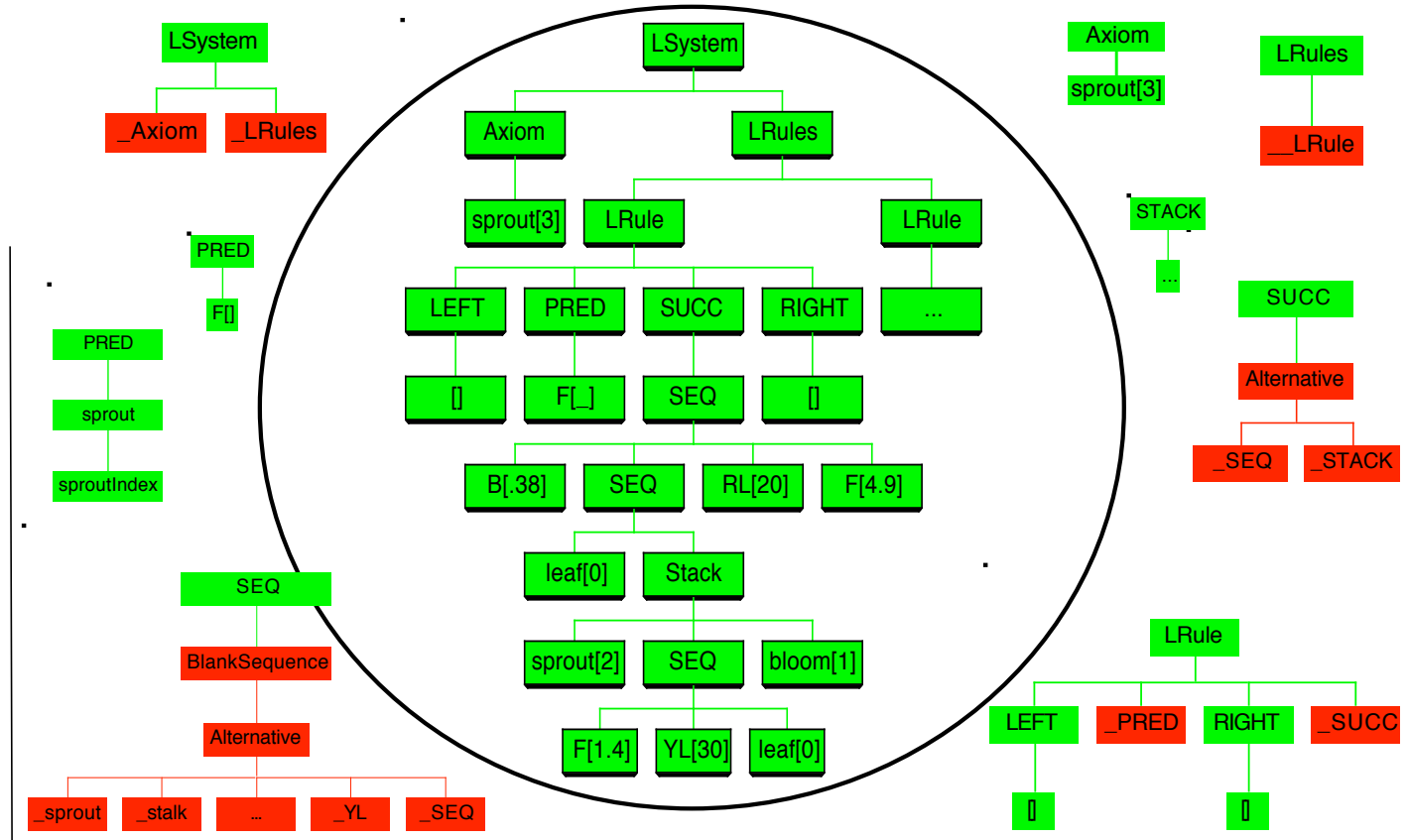
(9)

Generating program expressions

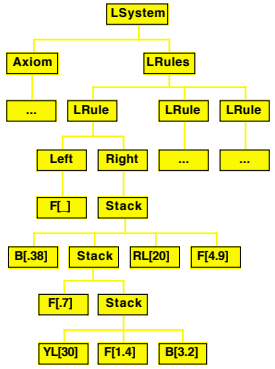


(10)

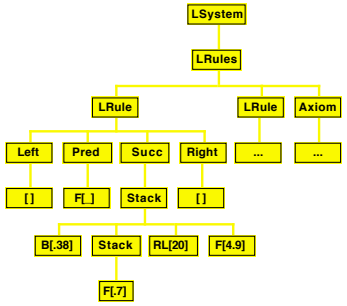
Generating program expressions



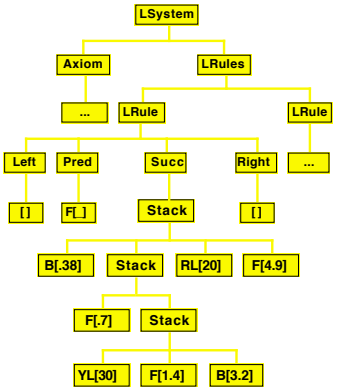
Population of Symbolic Expressions



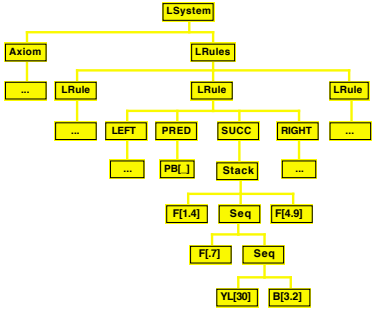
Program 1



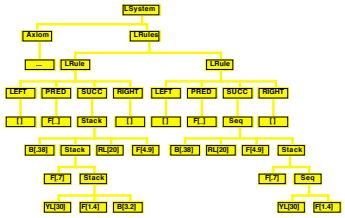
Program 2



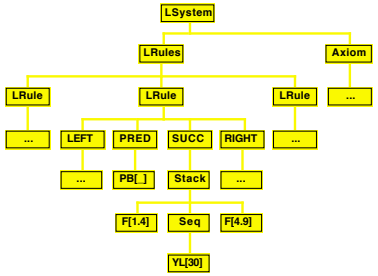
Program 3



Program 4

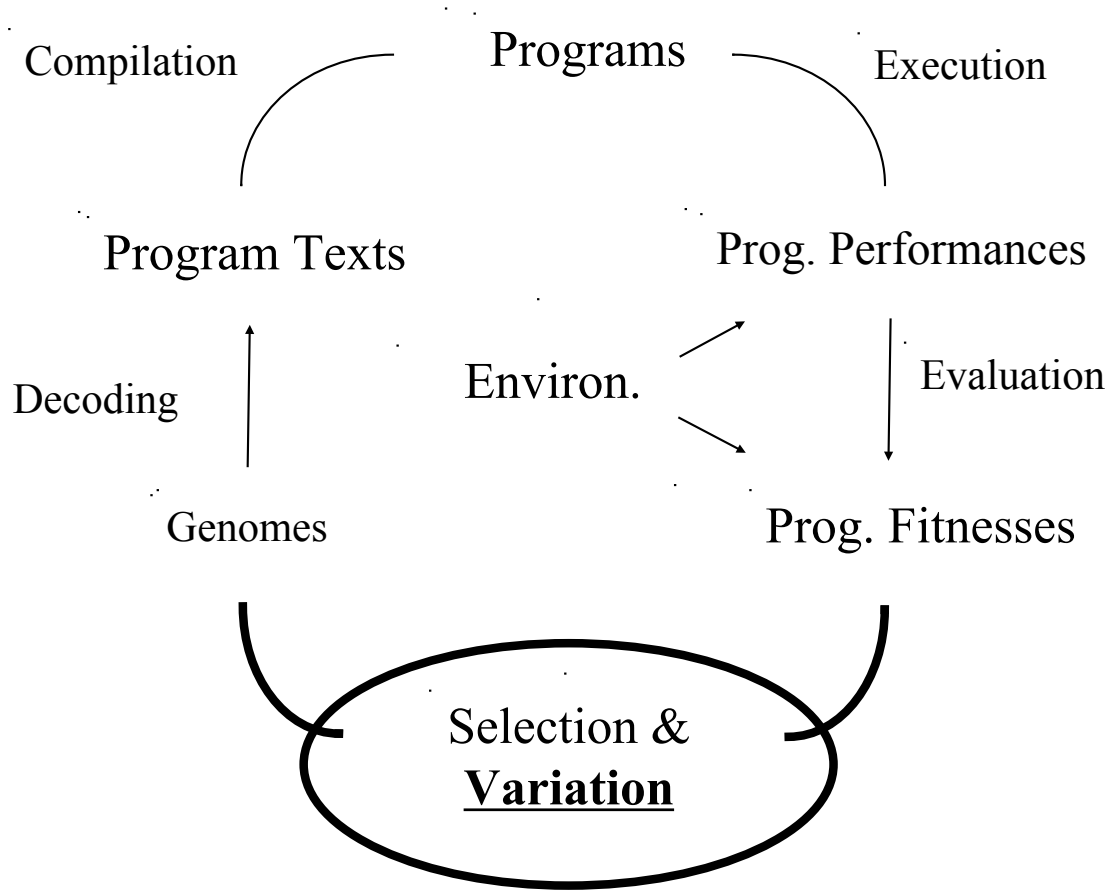


Program 5



Program 6

How to Diversify the Programs …

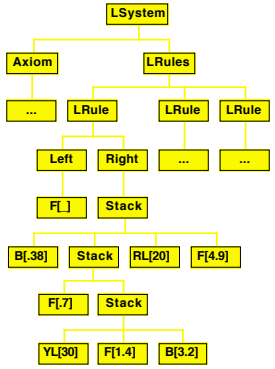


How to Diversify the Programs …

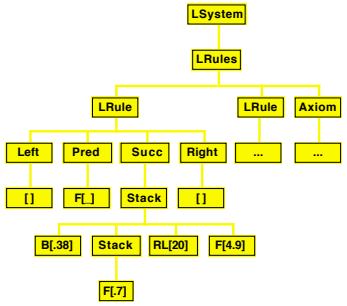
- **Genetic Operators on Symbolic Expressions**

- **Crossover**
- Mutation
  - Point Mutation
  - Tree Mutation
- Permutation
- Hoisting
- Deletion, Shrinking
- Expansion
- Duplication
- Encapsulation / Decapsulation

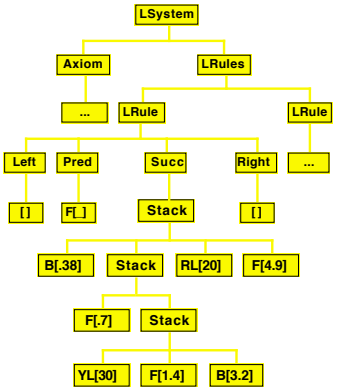
CROSSOVER on Symbolic Expressions



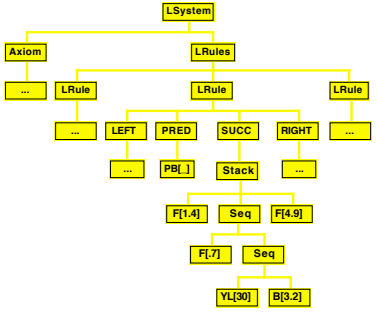
Program 1



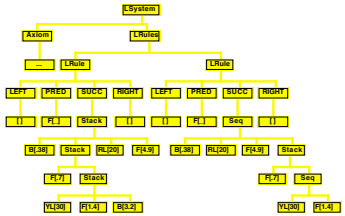
Program 2



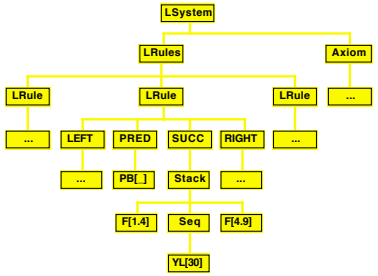
Program 3



Program 4

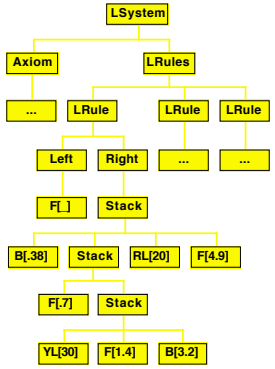


Program 5

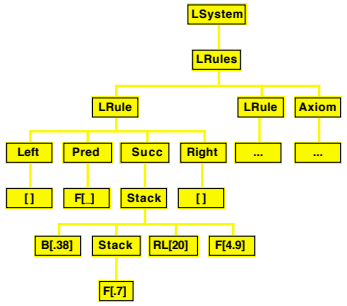


Program 6

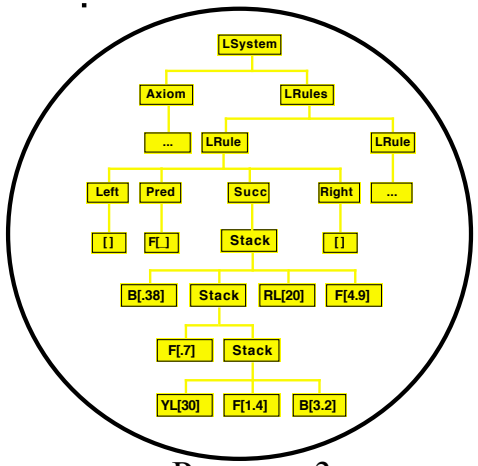
CROSSOVER on Symbolic Expressions



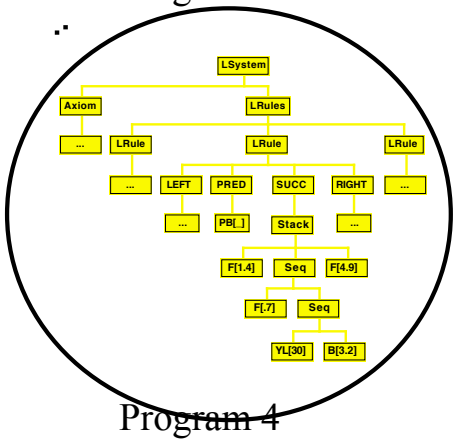
Program 1



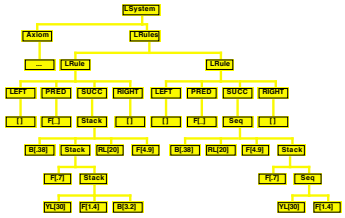
Program 2



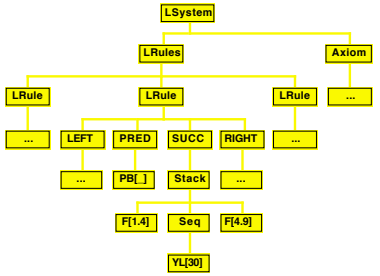
Program 3



Program 4

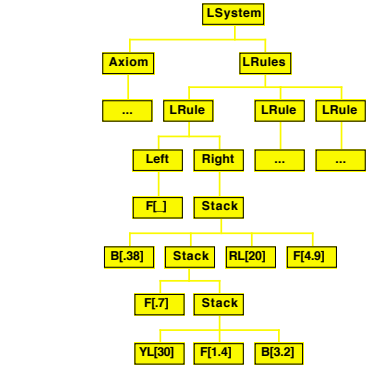


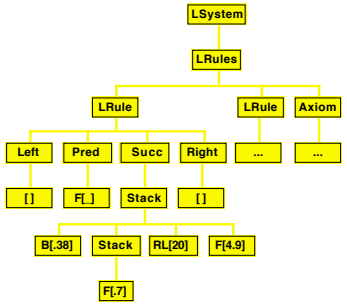
Program 5

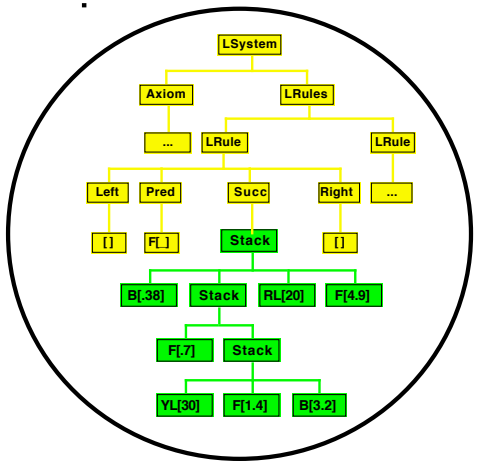


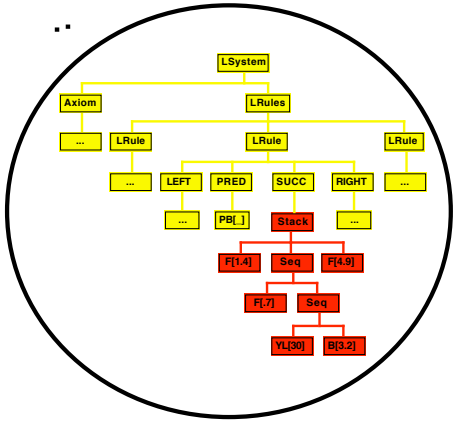
Program 6

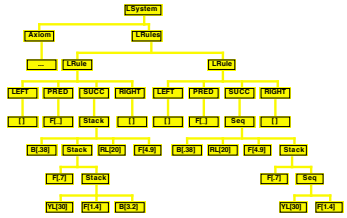
CROSSOVER on Symbolic Expressions

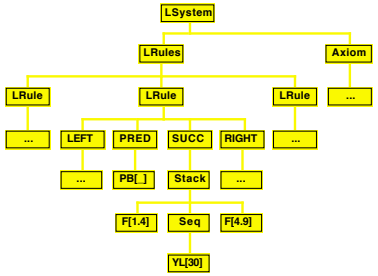




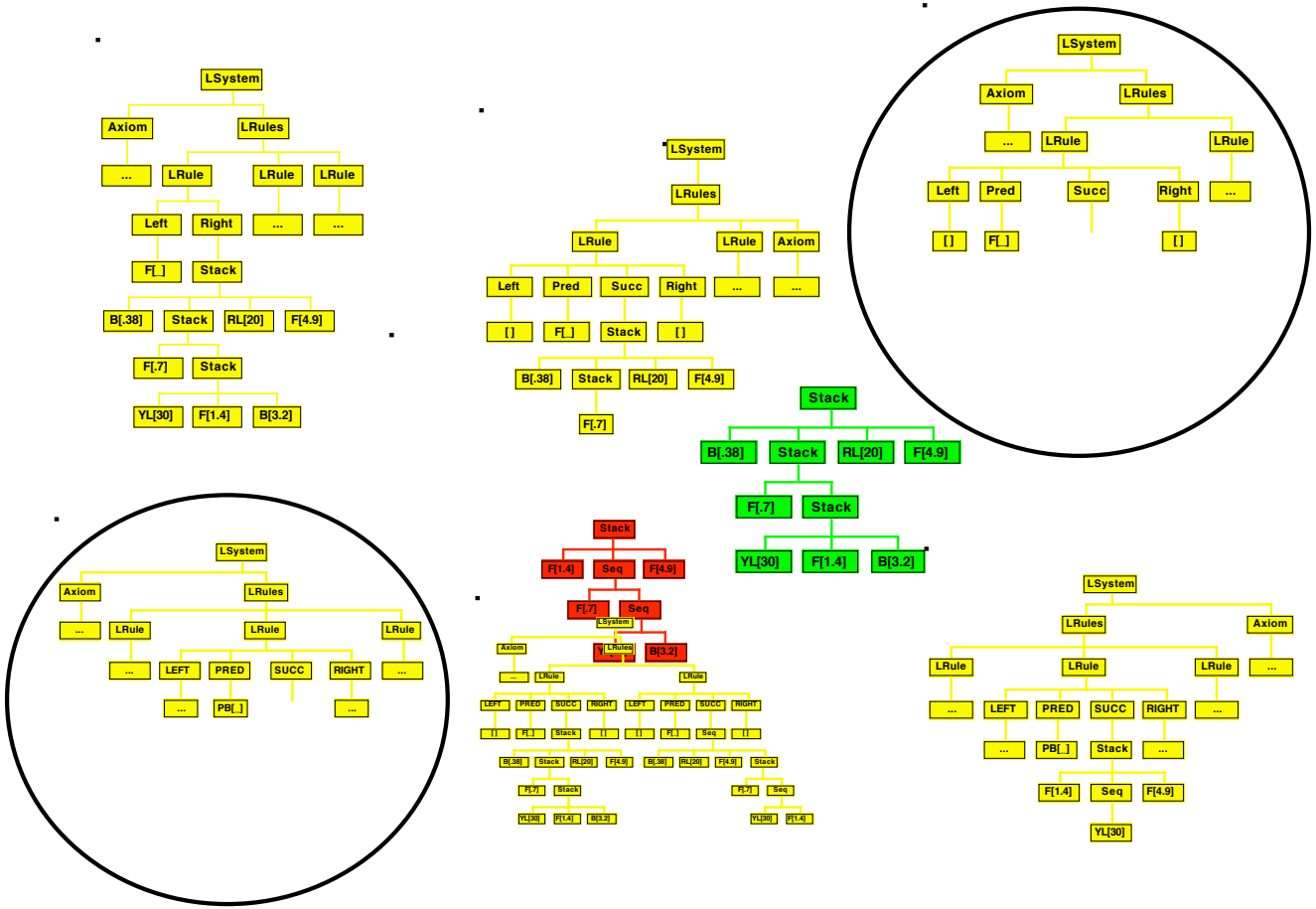




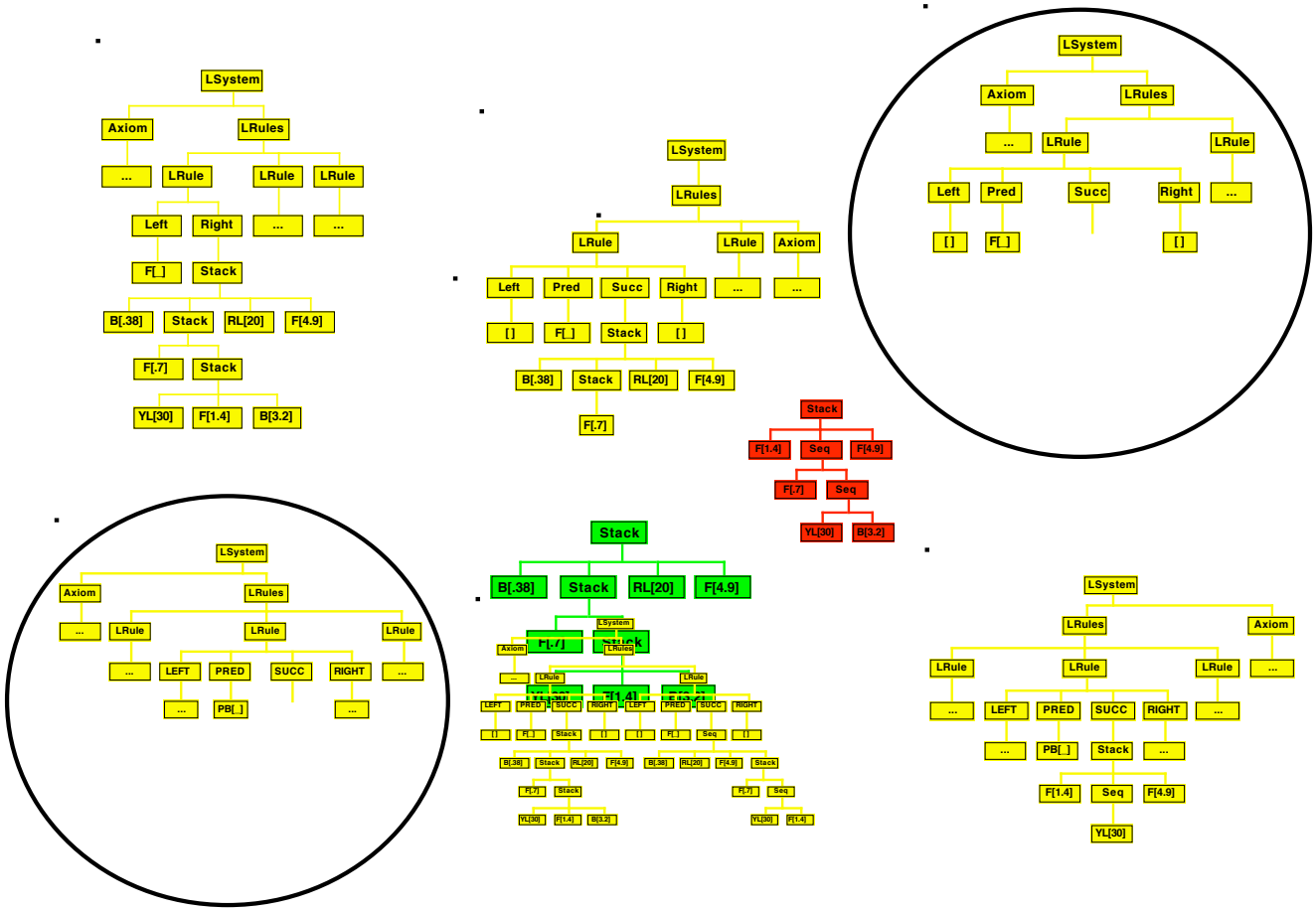




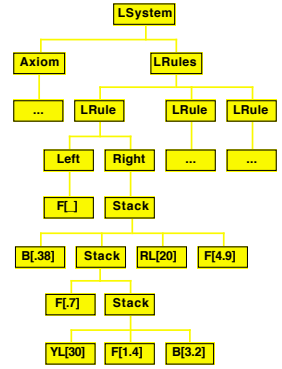
CROSSOVER on Symbolic Expressions

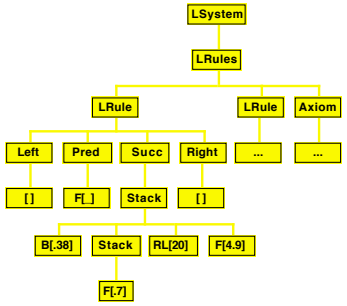


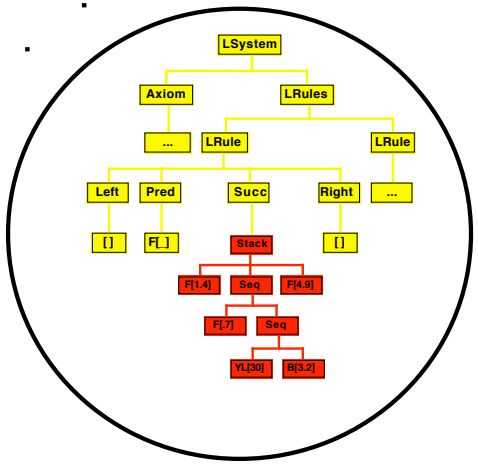
CROSSOVER on Symbolic Expressions

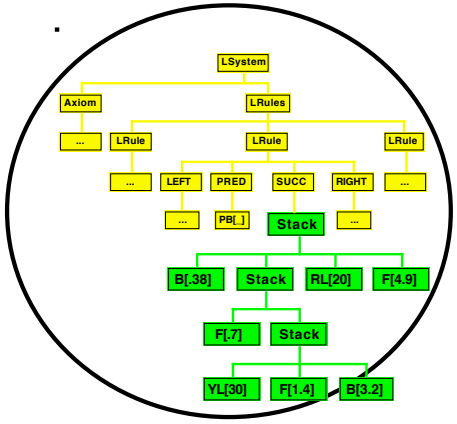


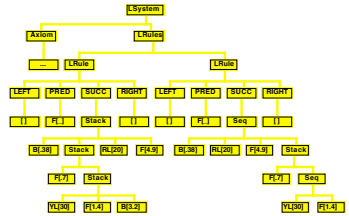
CROSSOVER on Symbolic Expressions

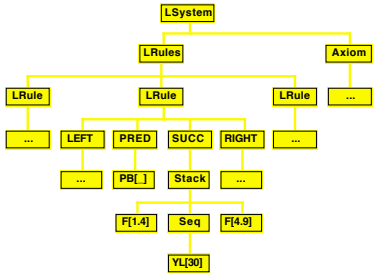




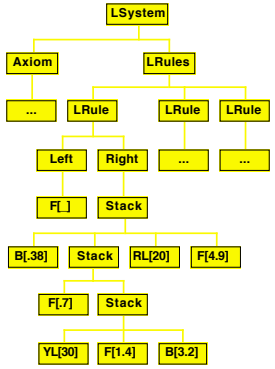




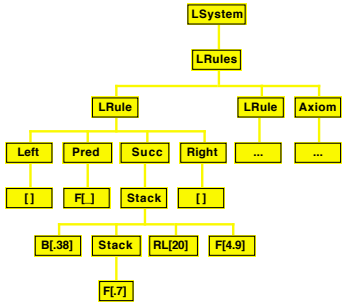




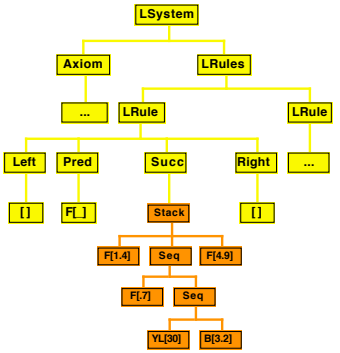
CROSSOVER on Symbolic Expressions



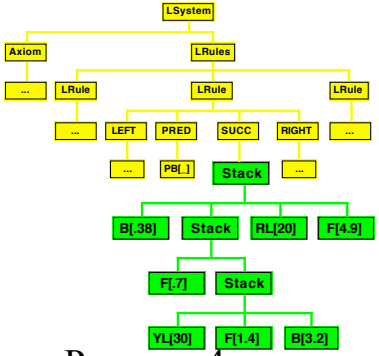
Program 1



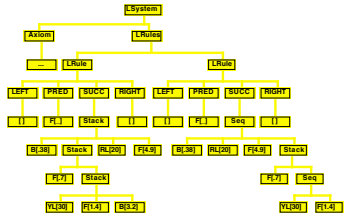
Program 2



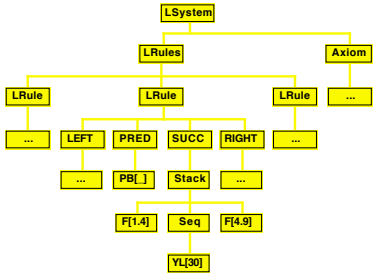
Program 3



Program 4



Program 5



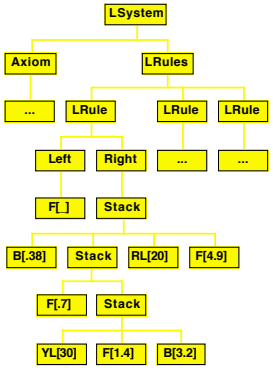
Program 6

How to Diversify the Programs … (2)

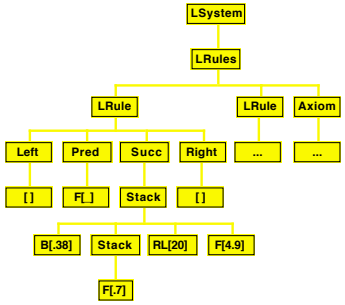
- **Genetic Operators on Symbolic Expressions**

- Crossover
- **Mutation**
  - Point Mutation
  - Tree Mutation
- Permutation
- Hoisting
- Deletion, Shrinking
- Expansion
- Duplication
- Encapsulation / Decapsulation

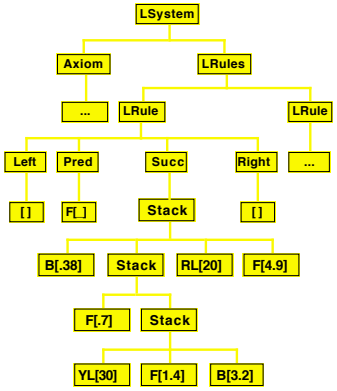
MUTATION on Symbolic Expressions



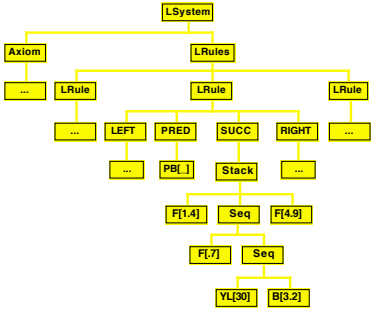
Program 1



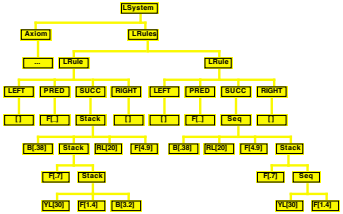
Program 2



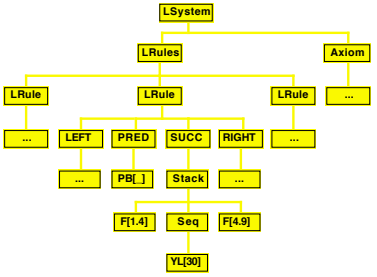
Program 3



Program 4

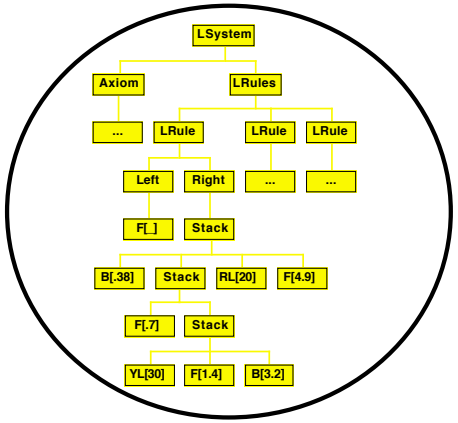


Program 5

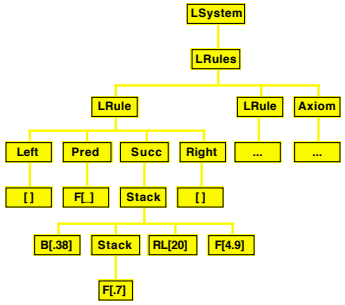


Program 6

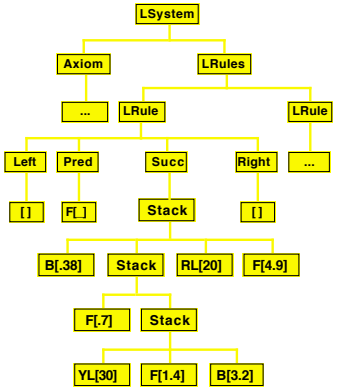
MUTATION on Symbolic Expressions



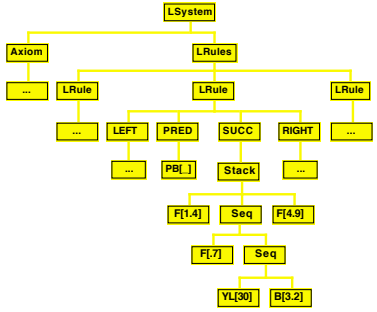
Program 1



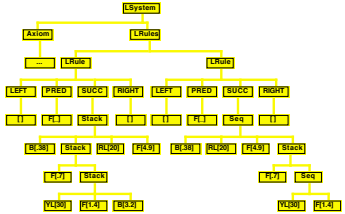
Program 2



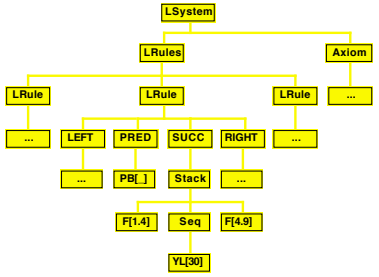
Program 3



Program 4

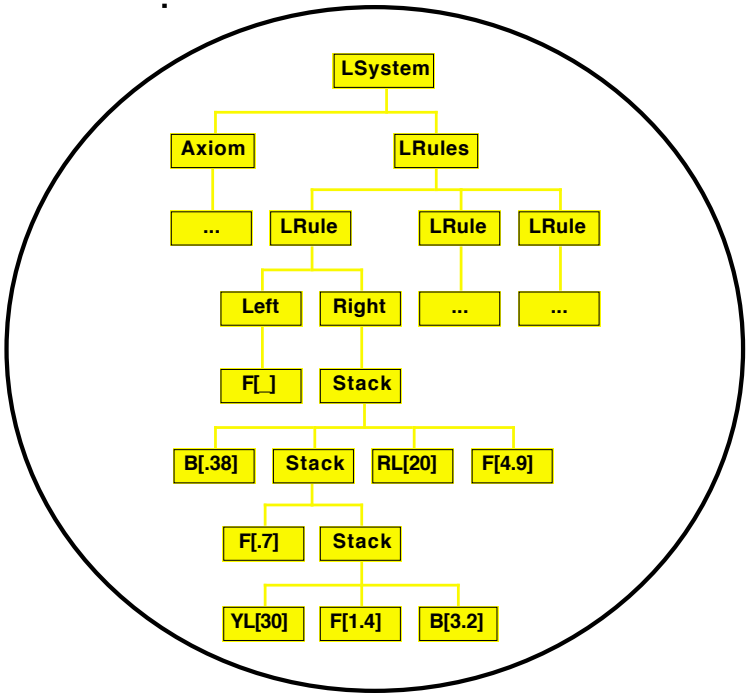


Program 5



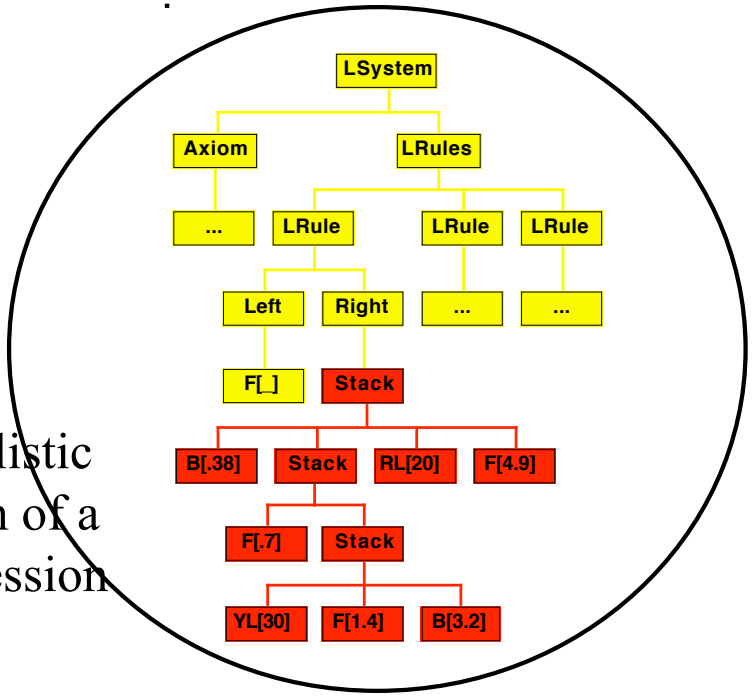
Program 6

MUTATION on a Symbolic Expression

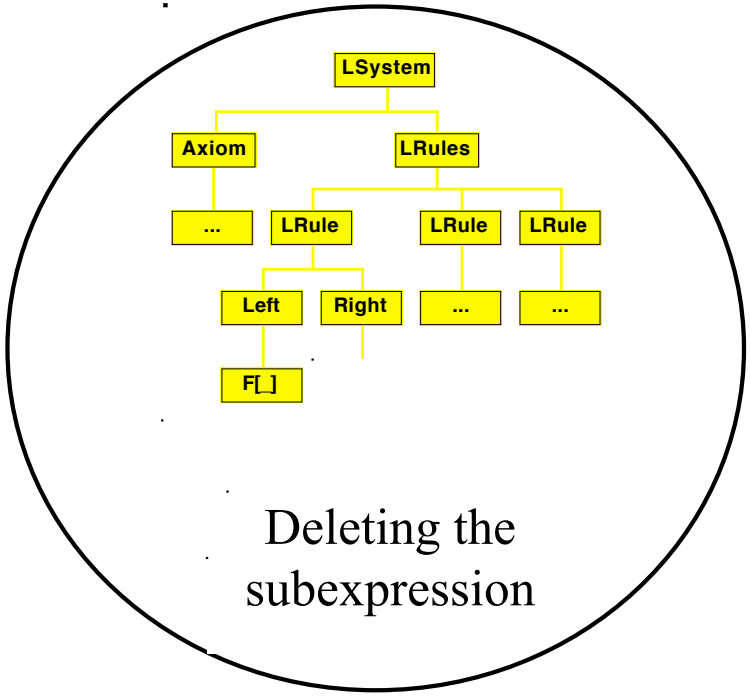


MUTATION on a Symbolic Expression

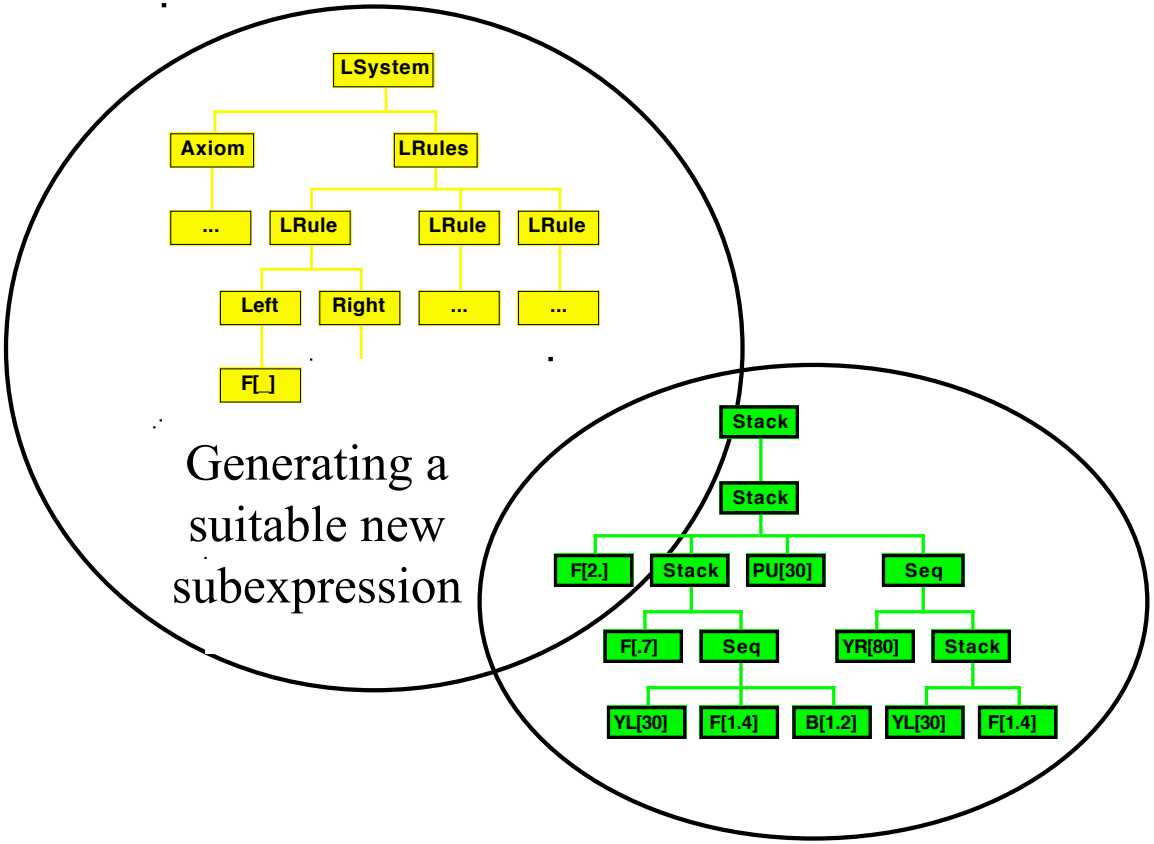
Probabilistic selection of a subexpression



MUTATION on a Symbolic Expression

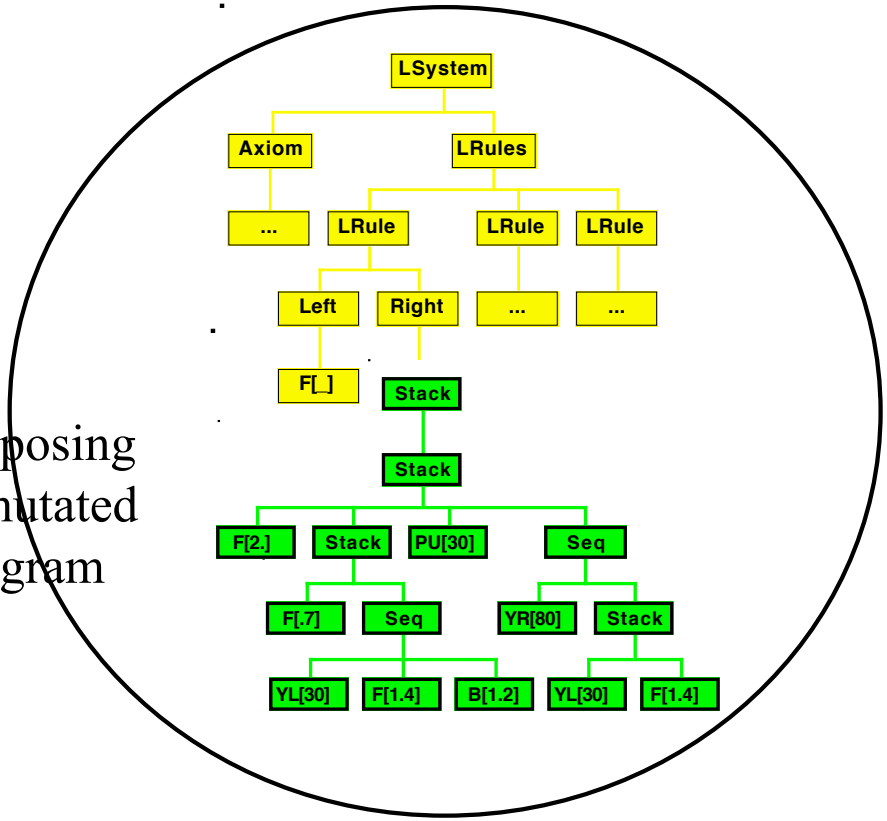


MUTATION on a Symbolic Expression

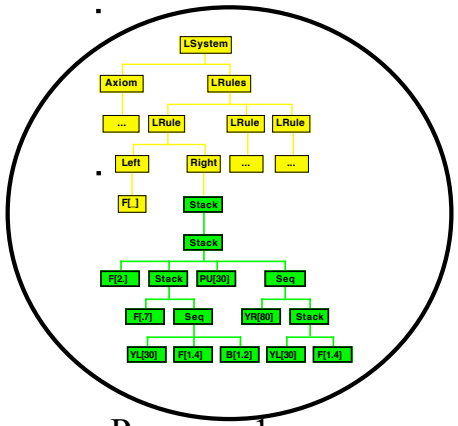


MUTATION on a Symbolic Expression

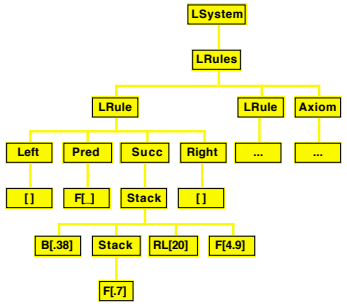
Composing the mutated program



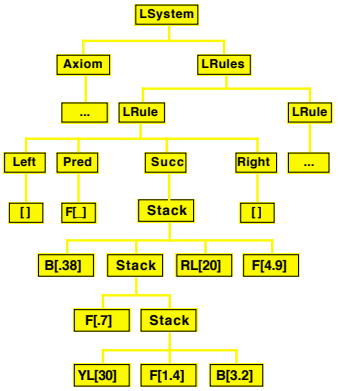
MUTATION on Symbolic Expressions



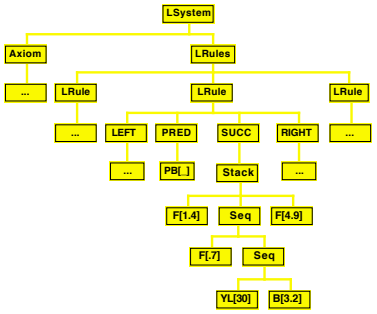
Program 1



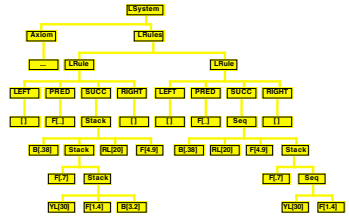
Program 2



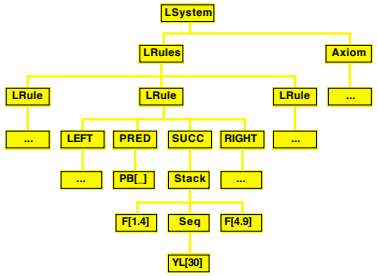
Program 3



Program 4

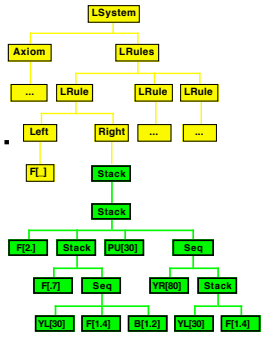


Program 5

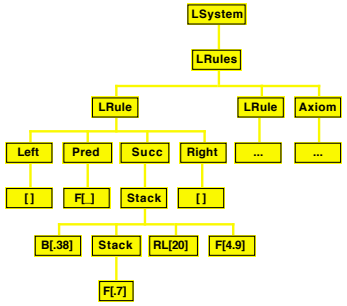


Program 6

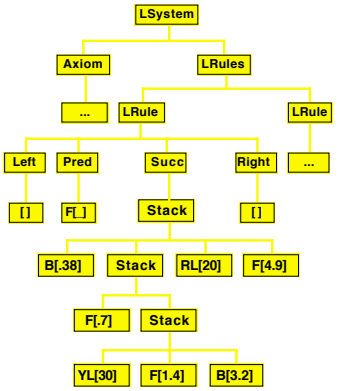
MUTATION on Symbolic Expressions



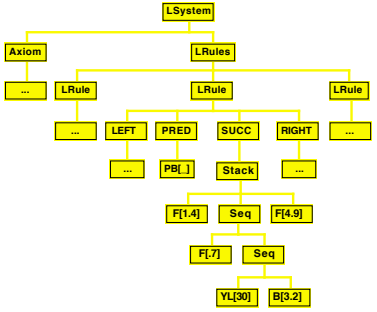
Program 1



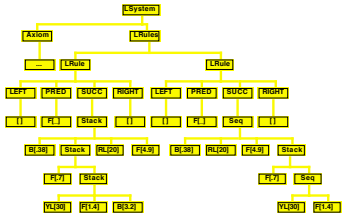
Program 2



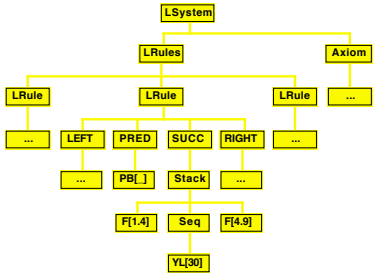
Program 3



Program 4

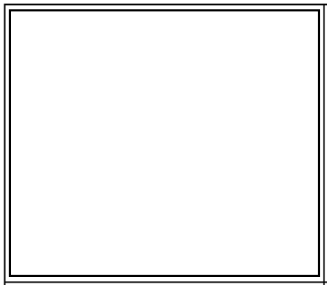
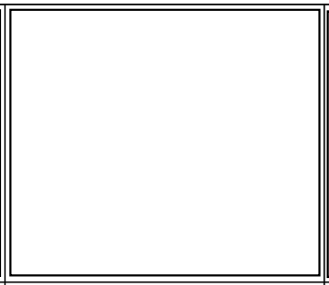
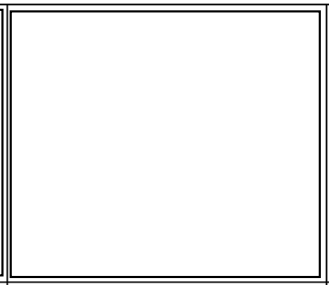
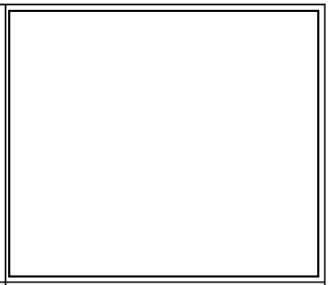
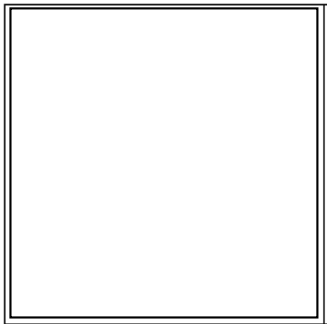
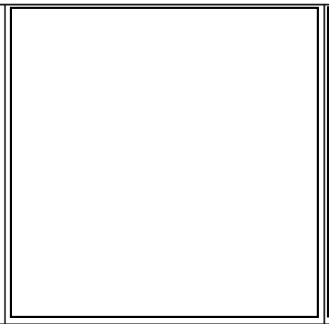
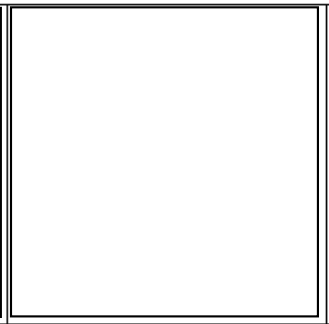
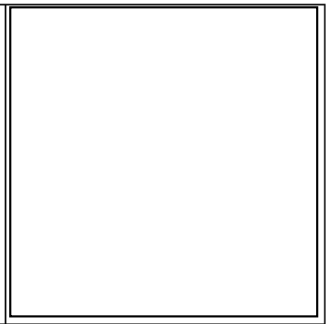


Program 5

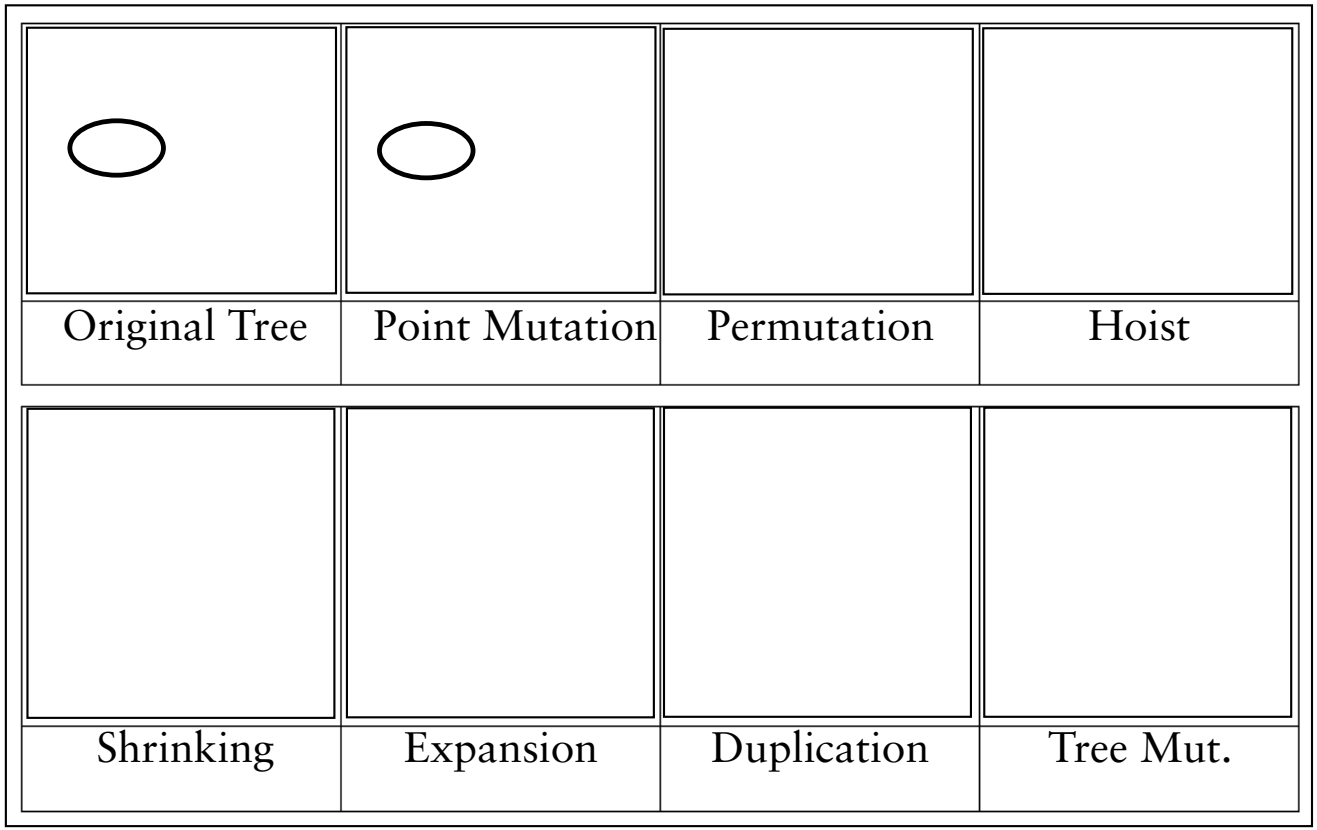


Program 6

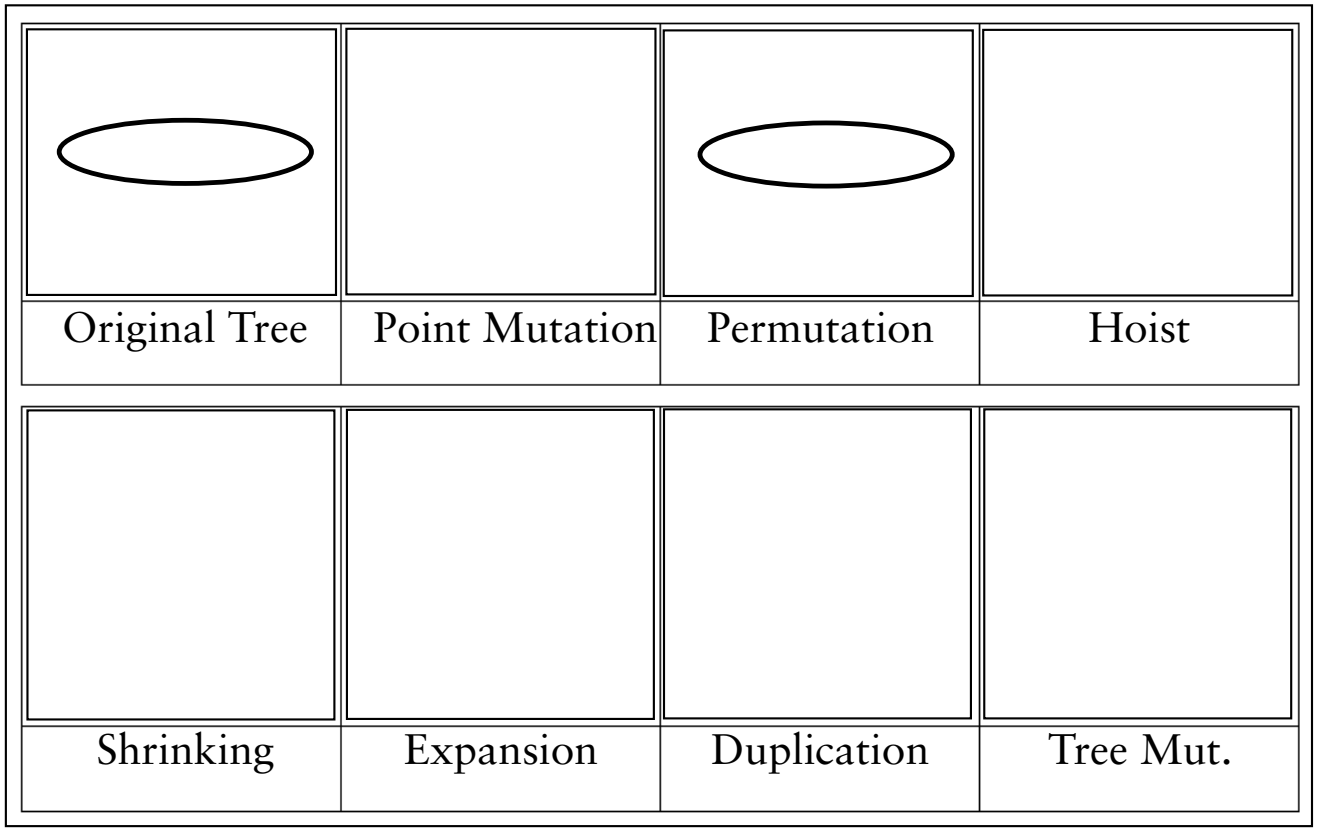
GP Mutation Operators

			
Original Tree	Point Mutation	Permutation	Hoist
			
Shrinking	Expansion	Duplication	Tree Mut.

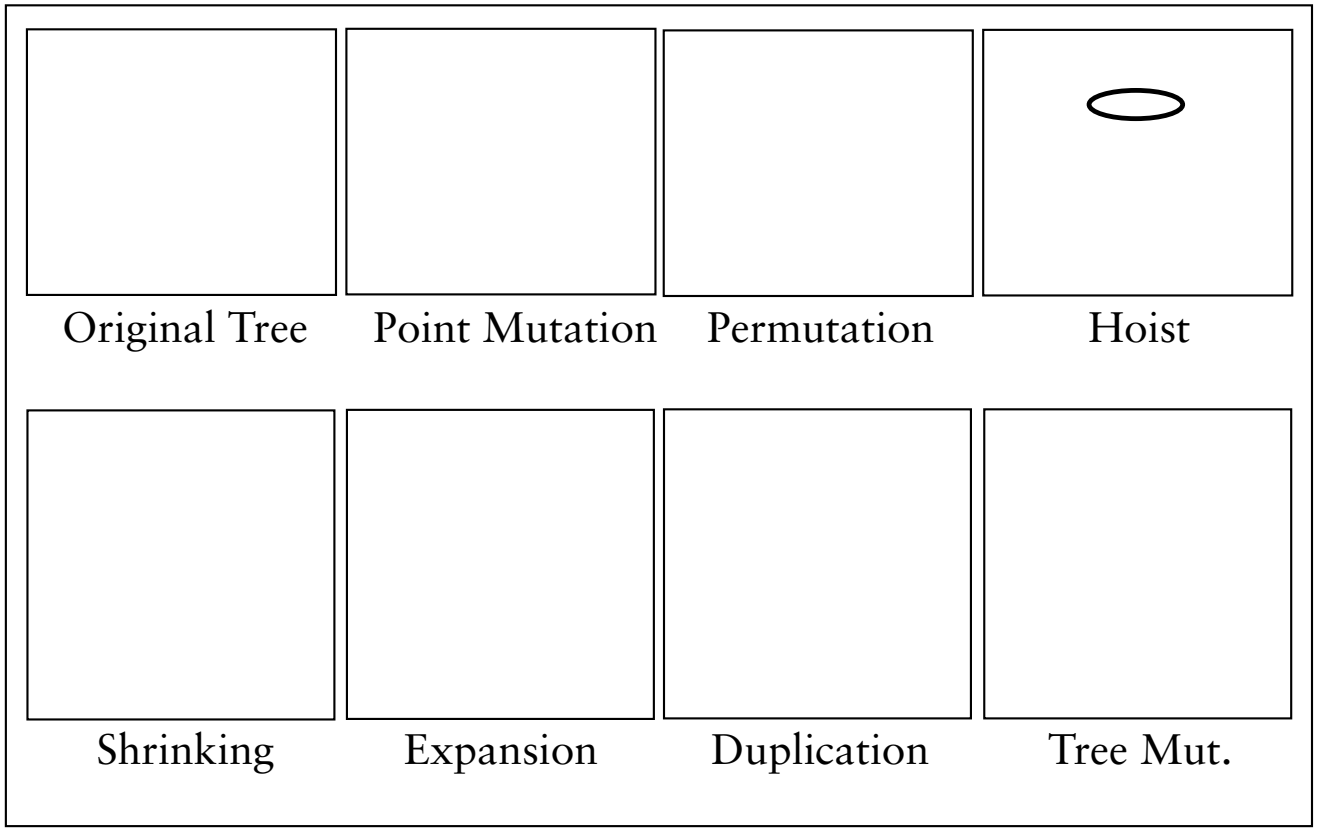
GP Mutation Operators



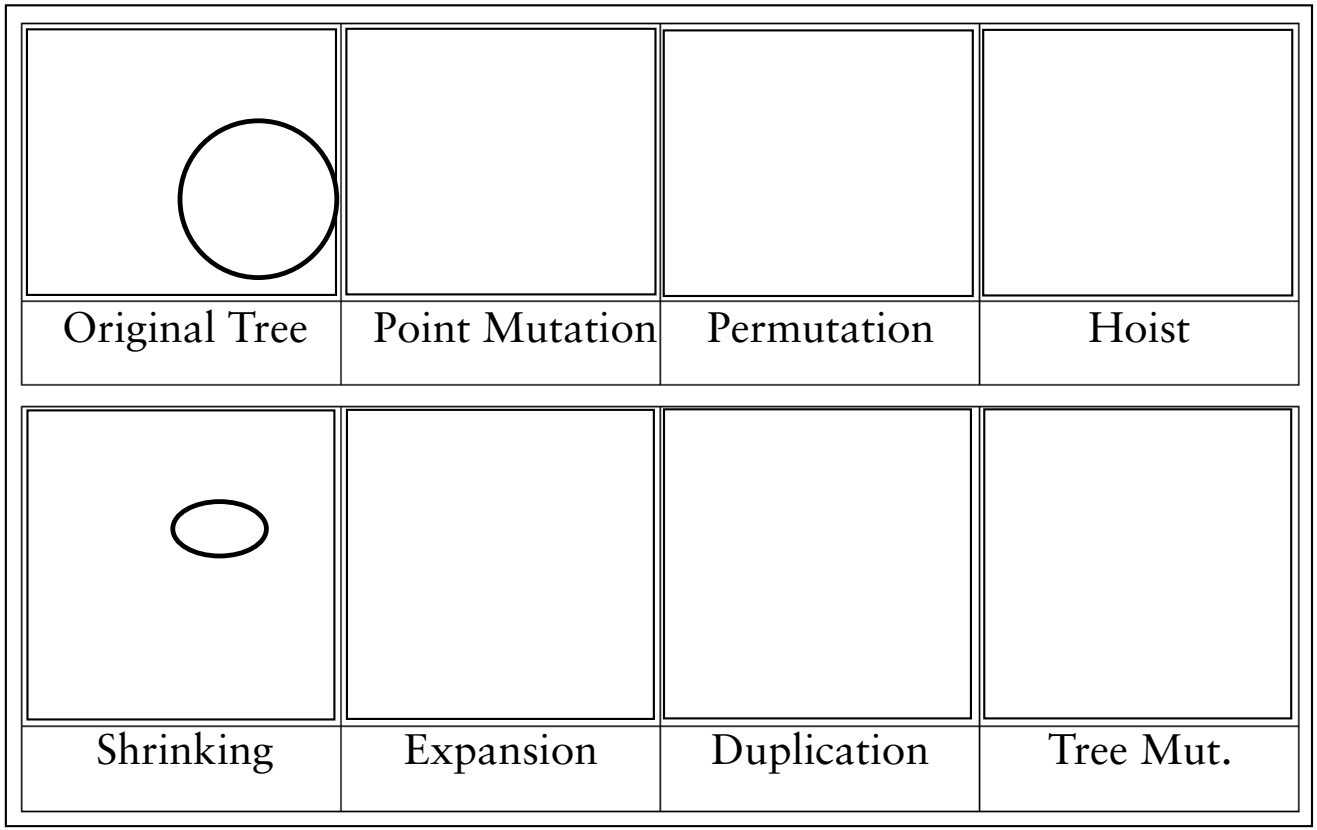
GP Mutation Operators



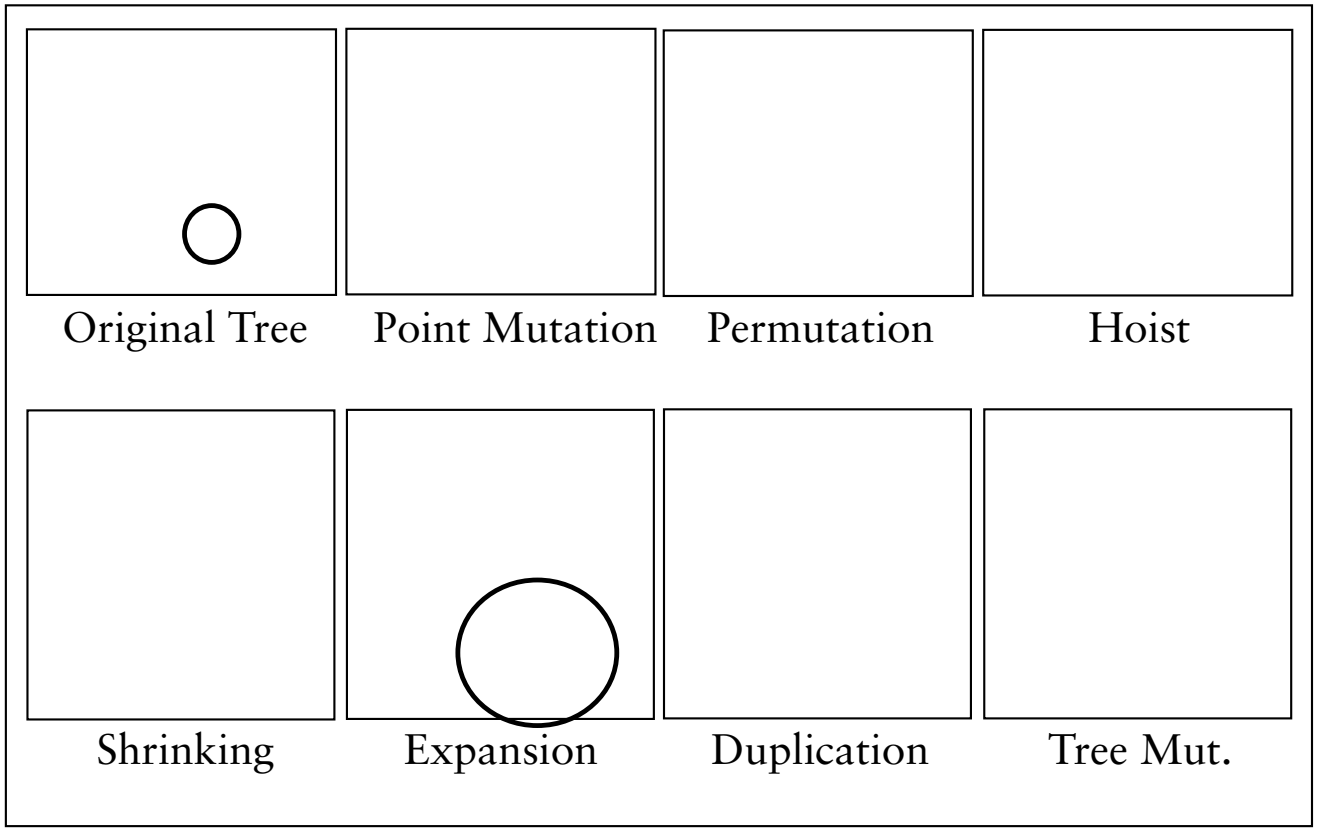
GP Mutation Operators



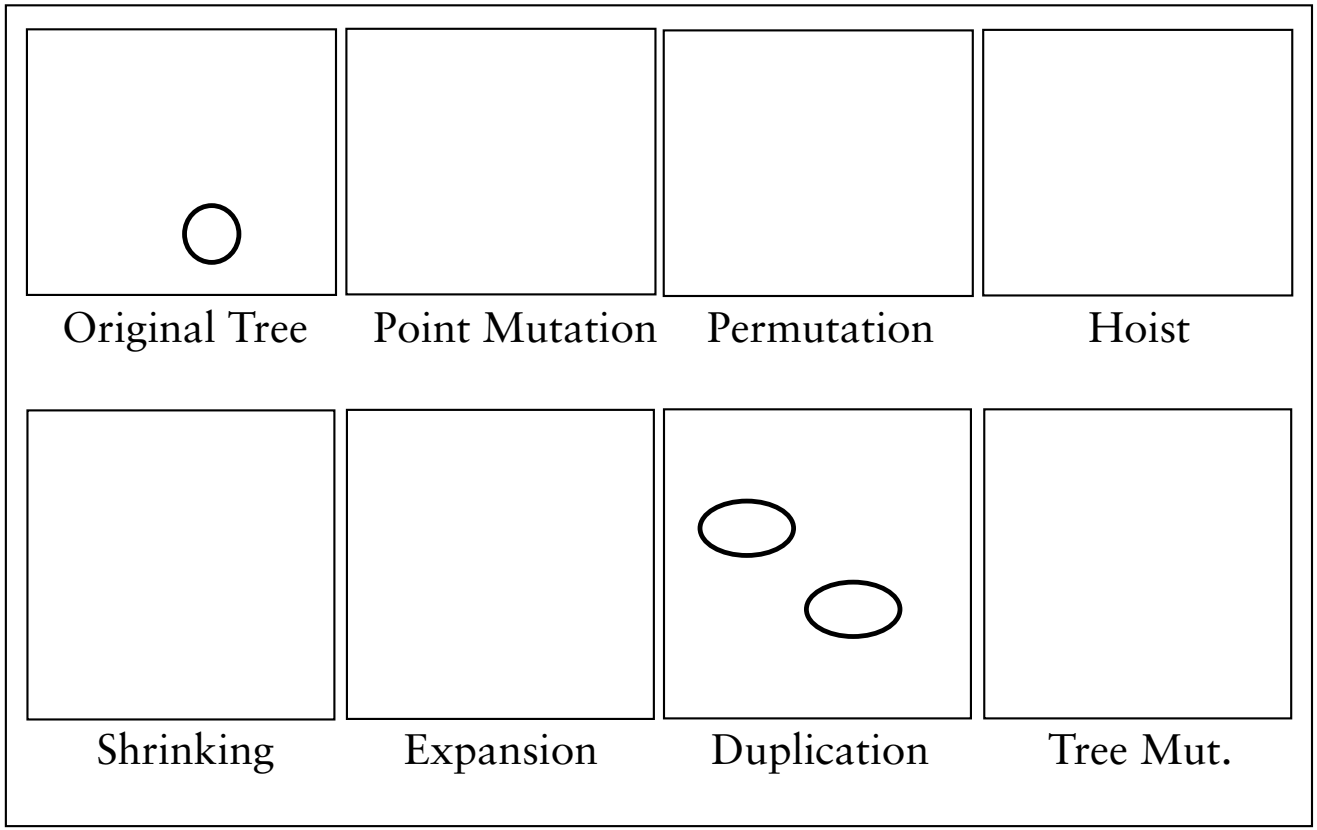
GP Mutation Operators



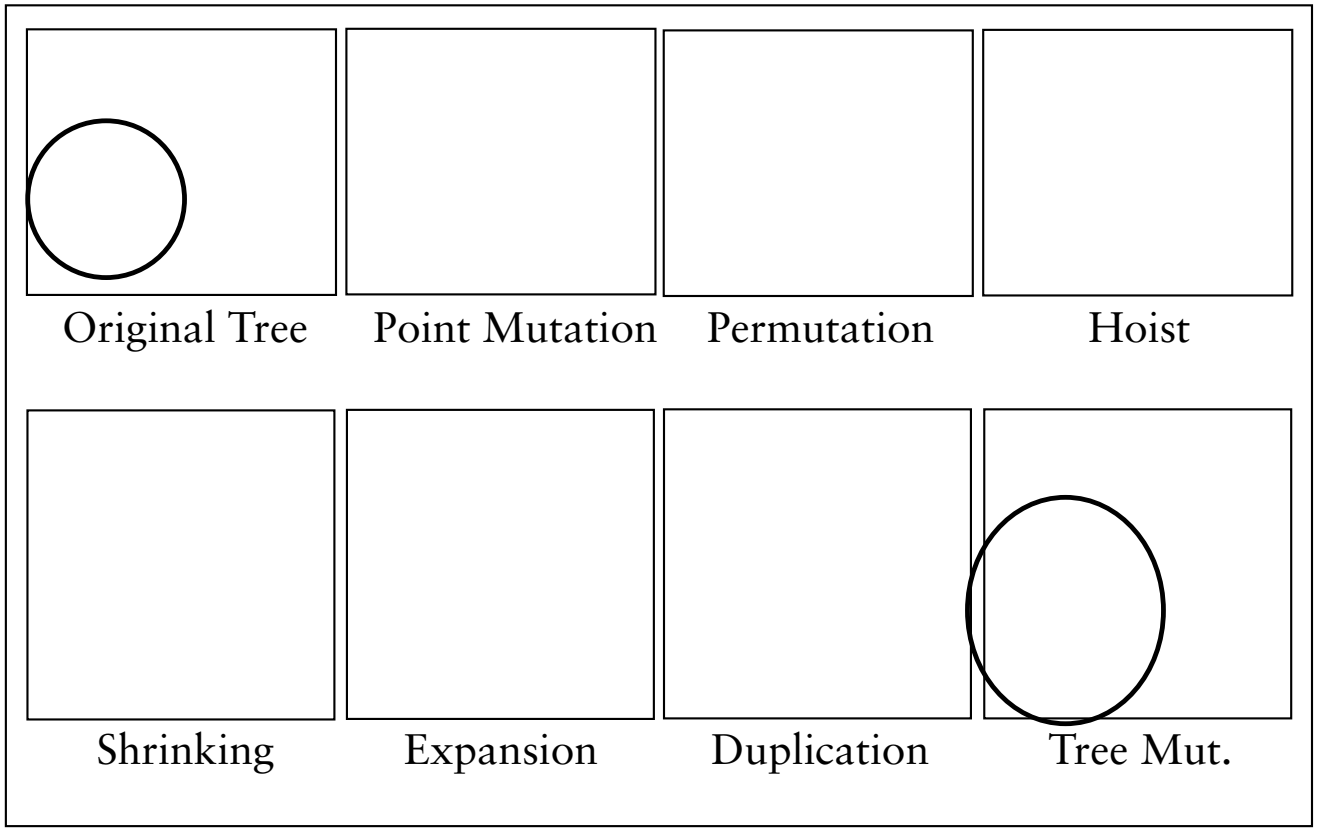
GP Mutation Operators



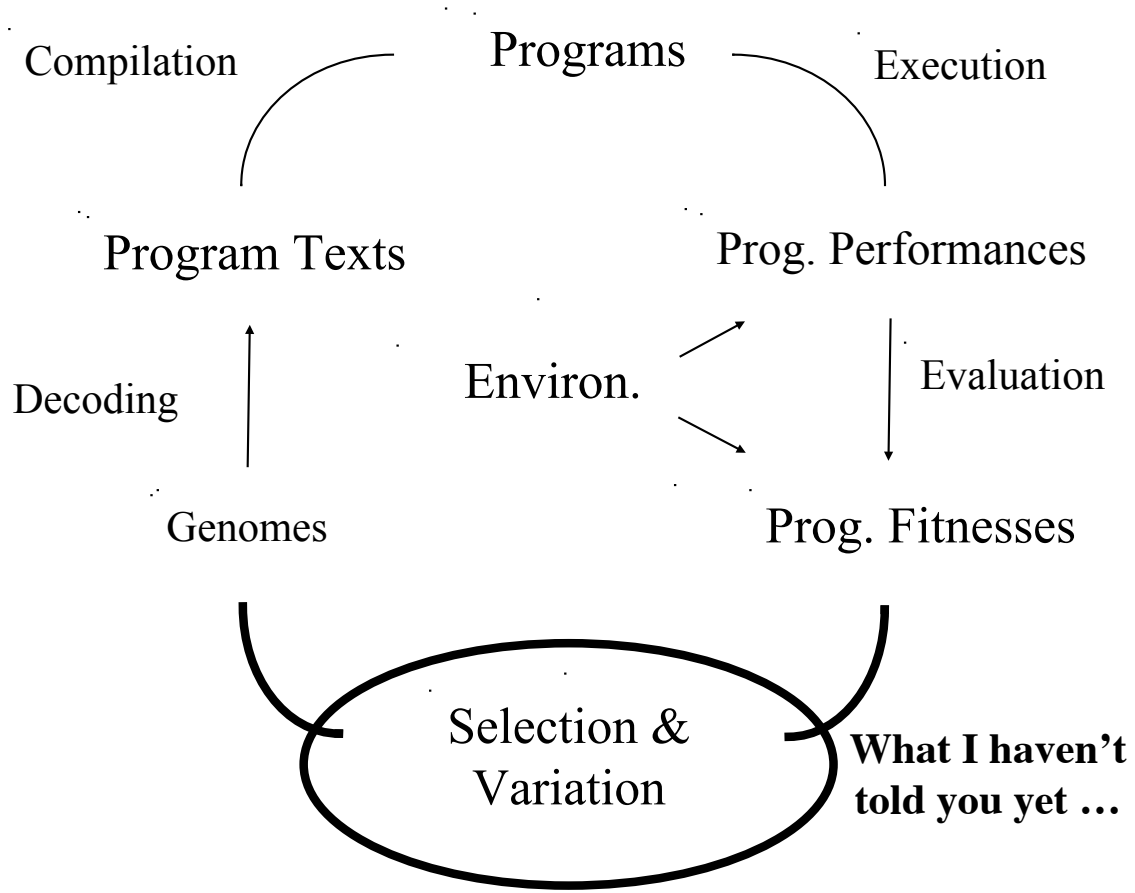
GP Mutation Operators



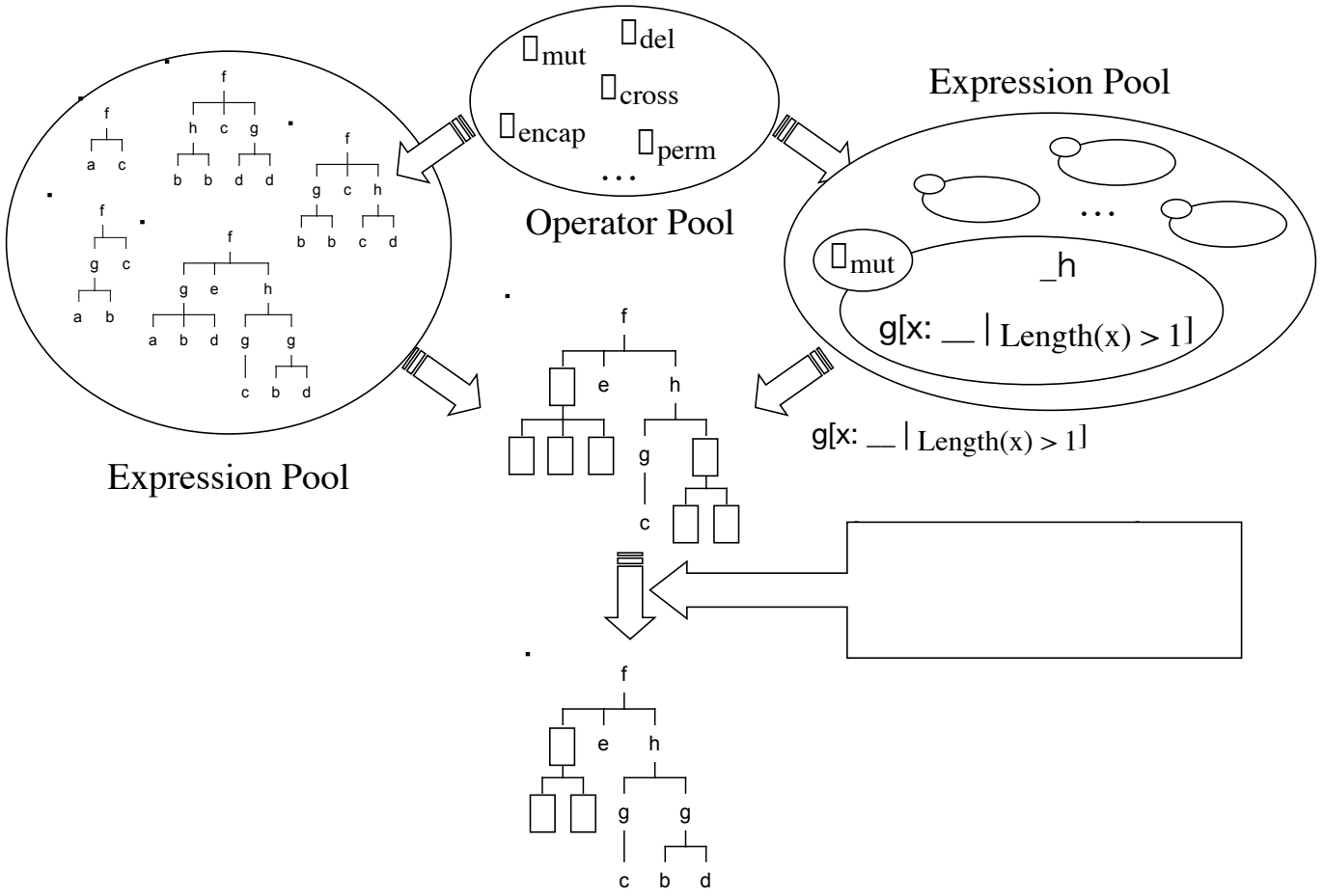
GP Mutation Operators



How to Diversify the Programs …



Variations of Expressions Using Templates



GP Operators: Selection and History

■ Operator Selection

Probability $\mathcal{P}(\omega)$ of selecting an operator $\omega \in \Omega$ from the operator set Ω:

$$\mathcal{P}(\omega) := \frac{\gamma(\omega)}{\sum_{\omega' \in \Omega} \gamma(\omega')}$$

■ History Function

- **Unary Operator**

$$\text{History}(s_i(t)) := (s_k(t-1), \omega)$$

- **Binary Operator**

$$\text{History}(s_i(t)) := \left(s_{k_1}(t-1), s_{k_2}(t-1), s_j(t), \omega_{\text{rec}}\right)$$

GP Operators: Weight Adaptation

■ Weight Adaptation

$\gamma(\omega, t)$ is the weight of operator ω in generation t.

$$\gamma(\omega, t) := \gamma(\omega, t-1) \cdot \Delta\gamma(\omega, t) \cdot \gamma_{\text{decay}}(\omega)$$

Average fitness difference: $\Delta\gamma(\omega, t)$.
Weight decay constant: $0 \leq \gamma_{\text{decay}}(\omega) \leq 1$.

GP Operators: Weight Adaptation (2)

- **$\Delta\gamma$ for Unary Operators**

$$\Delta\gamma(\omega, t) := \frac{1}{|P(\omega,t)|} \cdot \sum_{s \in P(\omega,t)} \frac{\tau(s)}{\tau(\text{History}(s)_1)}$$

$$P(\omega, t) := \{s \mid \text{History}(s)_2 = \omega\} \subseteq P(t)$$

$P(\omega, t)$ is the set of all structures within a population $P(t)$, which are offspring from parents in $P(t-1)$ using operator ω.

- **$\Delta\gamma$ for Binary Operators**

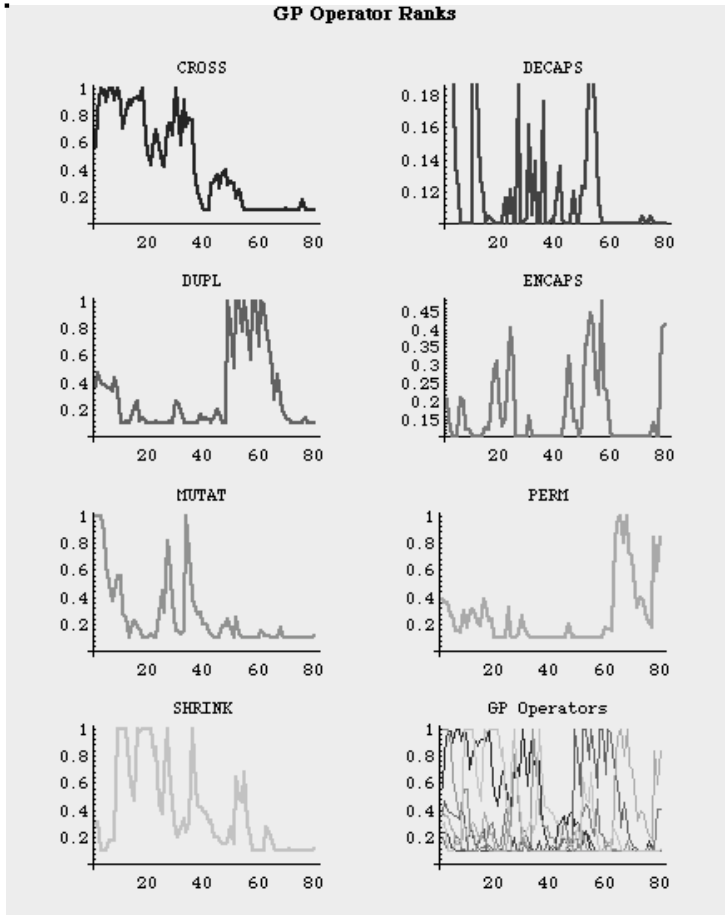
$$\Delta\gamma(\omega, t) := \frac{1}{2 \cdot |P(\omega,t)|} \cdot \sum_{s \in P(\omega,t)} \tau^*(s)$$

$$\tau^*(s) := \text{Max}\left(\frac{\tau(s)}{\tau(p_1(s))}, \frac{\tau(b(s))}{\tau(p_2(s))} \right)$$

Parents: $p_i(s) := \text{History}(s)_i$

Offspring: $b(s) := \text{History}(s)_3$

AntTracker: Adaptive Operator Ranks



[Jacob 2001, Ch. 9]

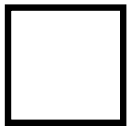
Evolution of Fractals

Towards a better understanding of replacement systems and their evolution

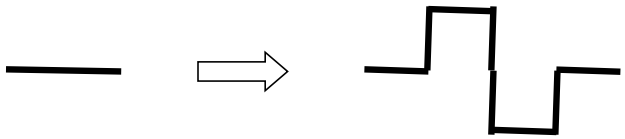
Jing Yu

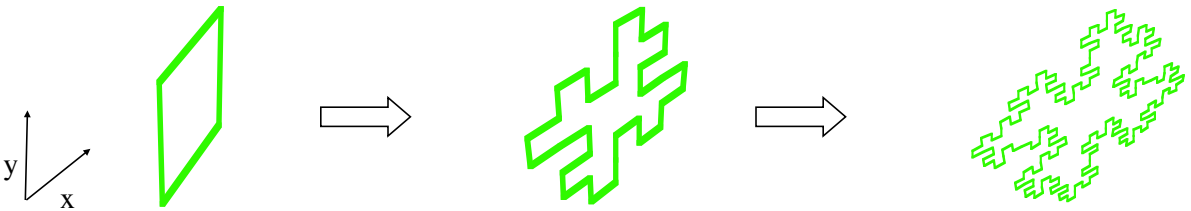
Fractals and Lindenmayer Systems

Axiom:



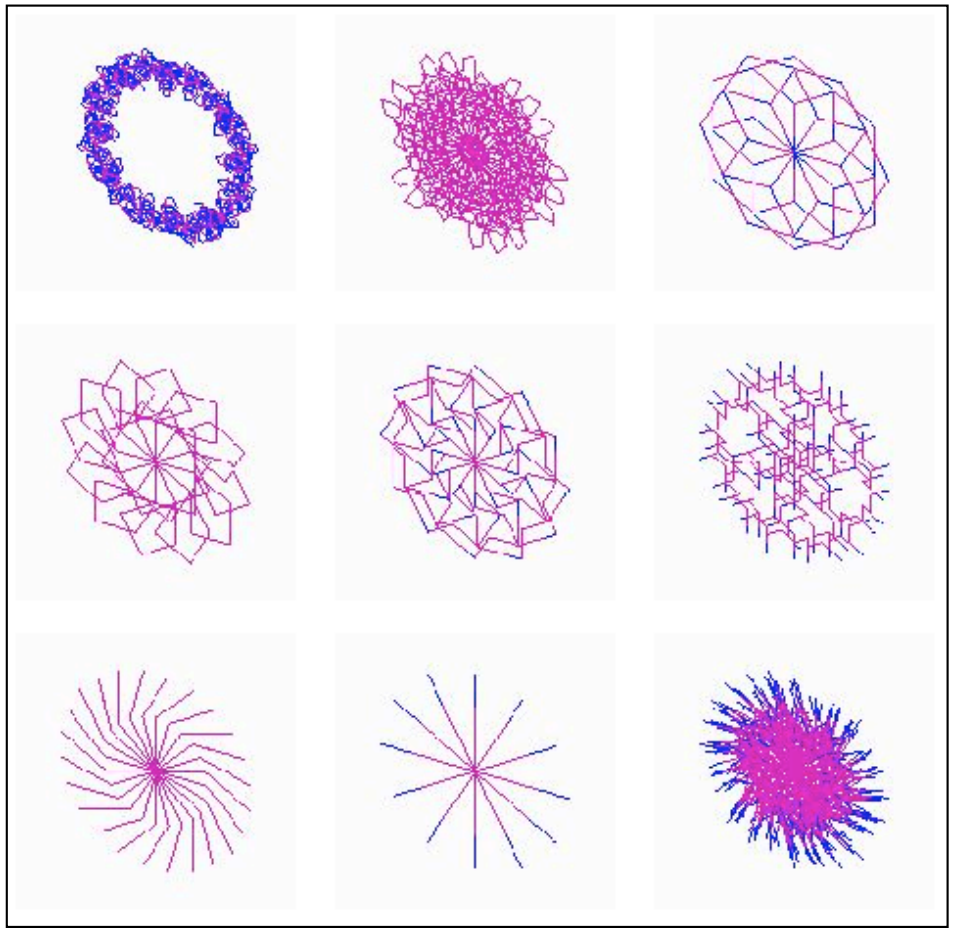
Replacement Rule:



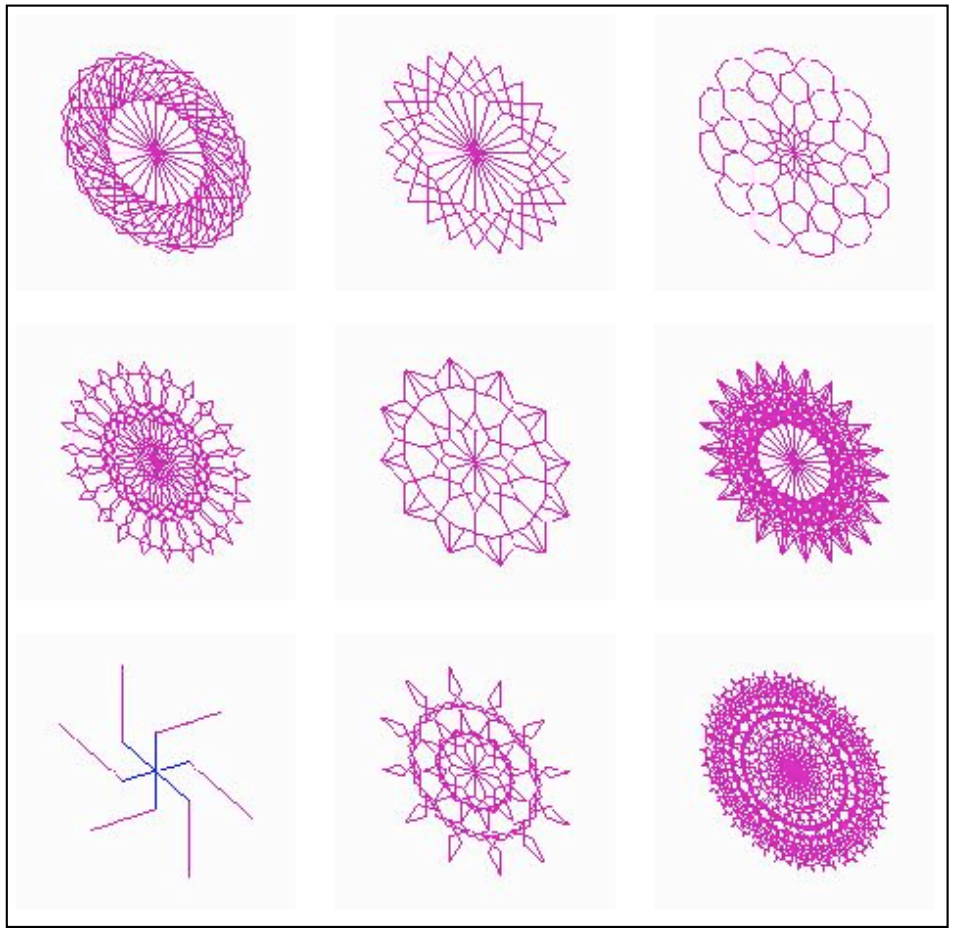


The Quadratic Koch Island

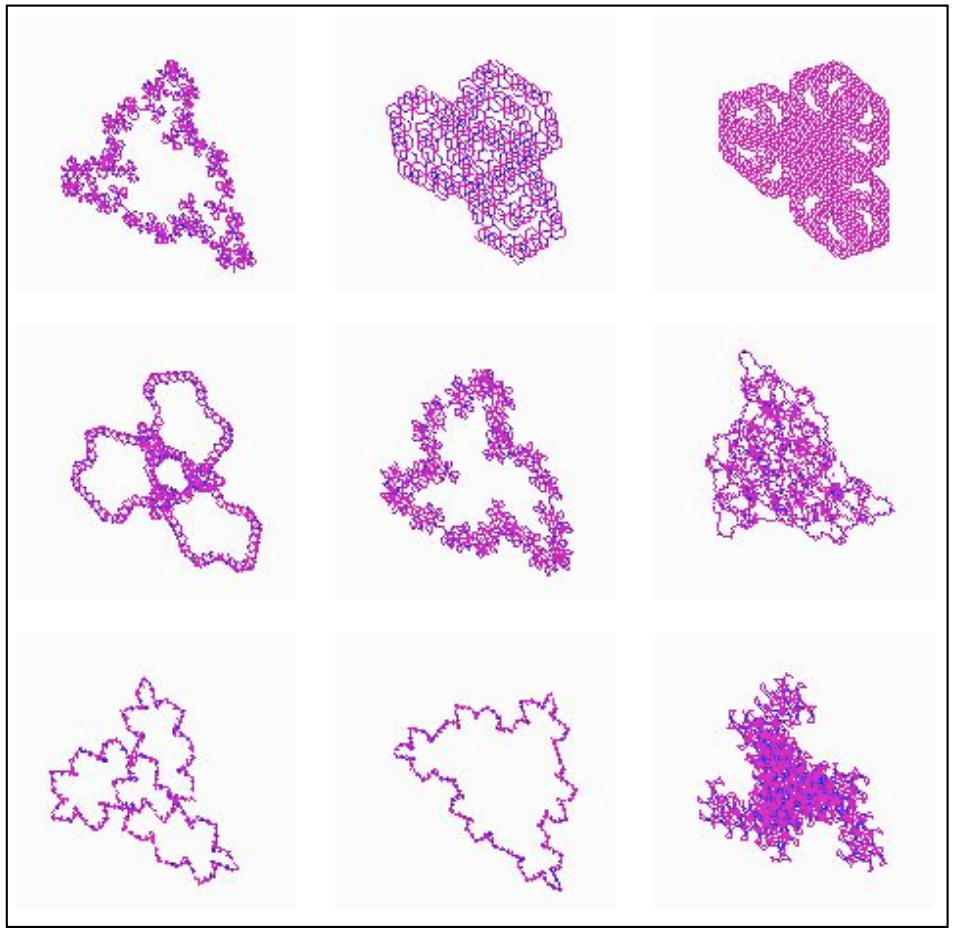
Gallery of Evolved Fractals



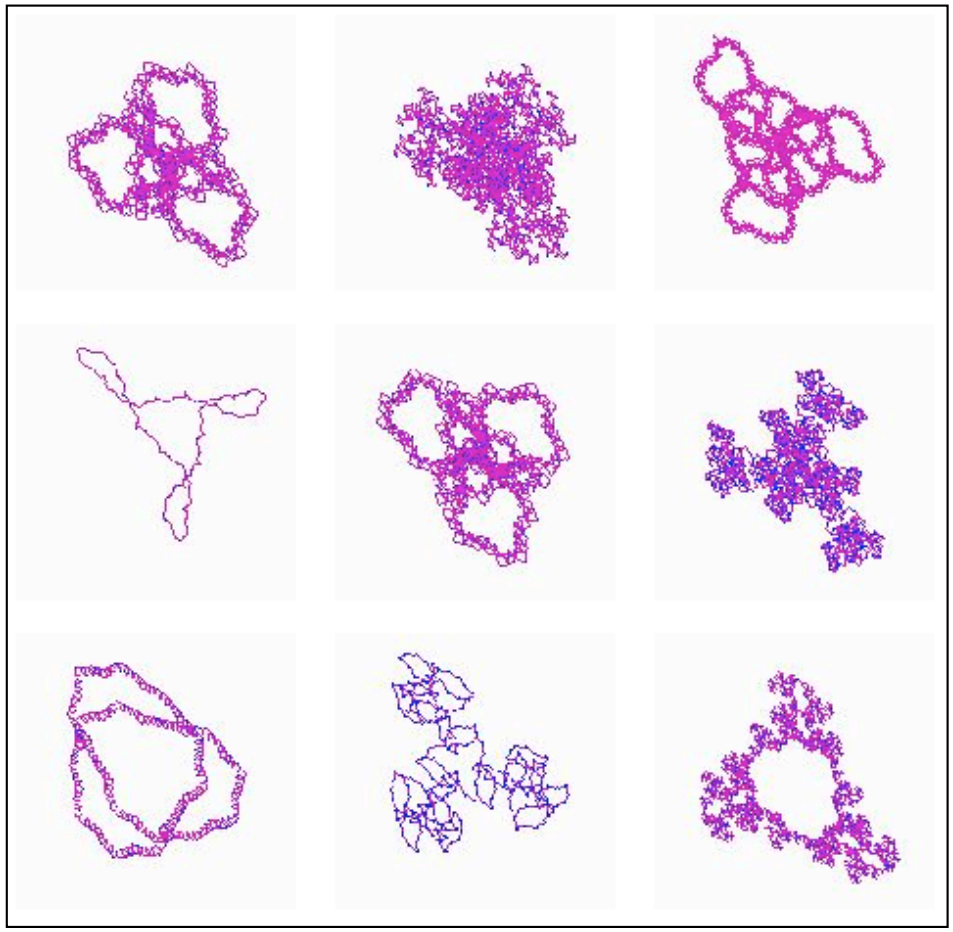
Gallery of Evolved Fractals



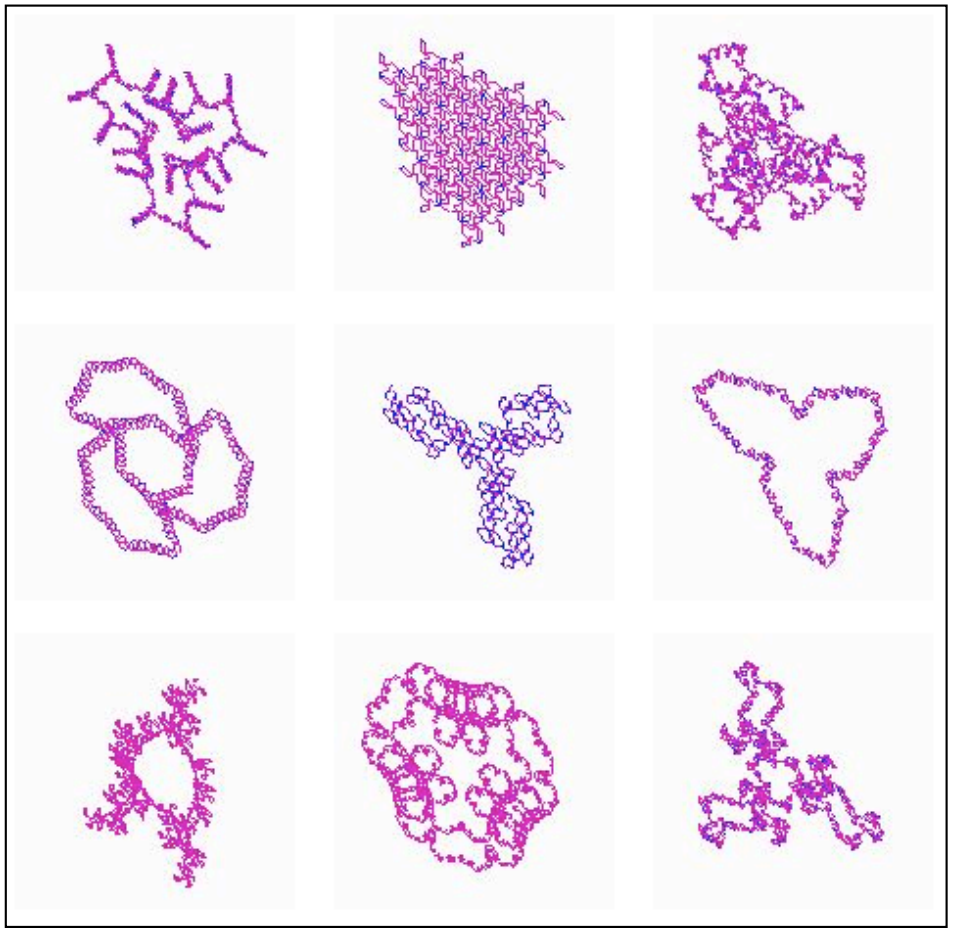
Gallery of Evolved Fractals



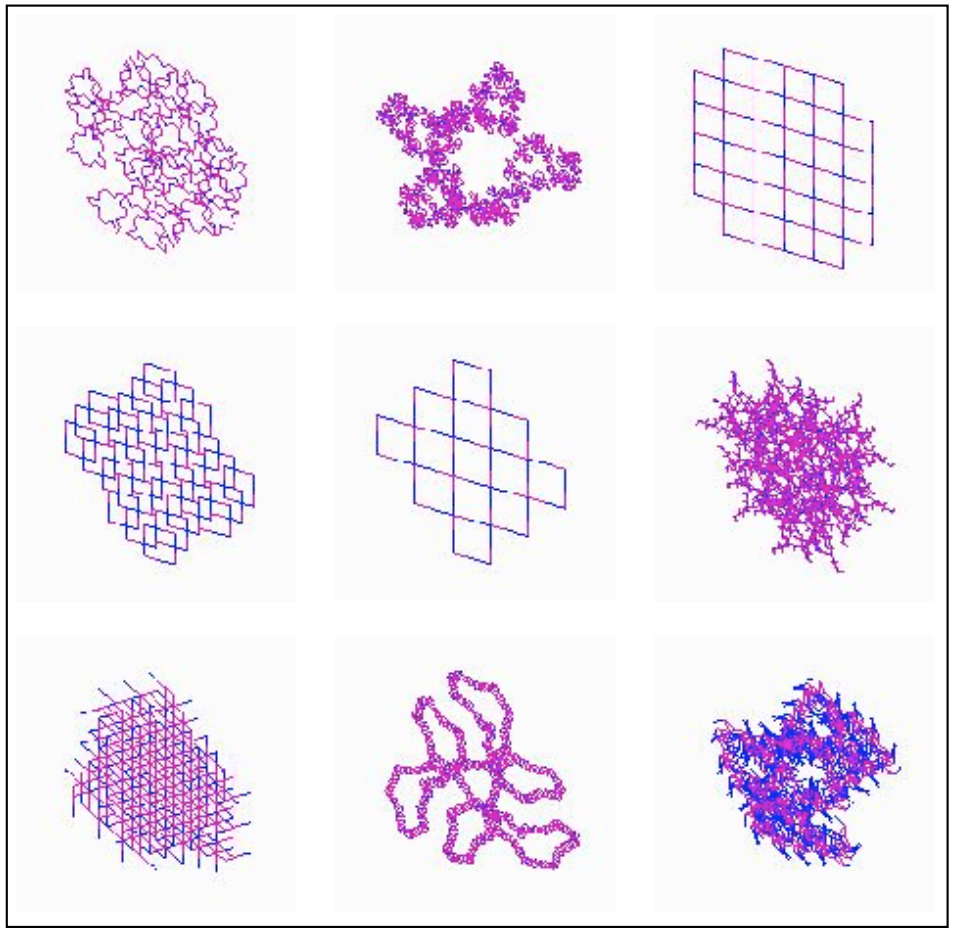
Gallery of Evolved Fractals



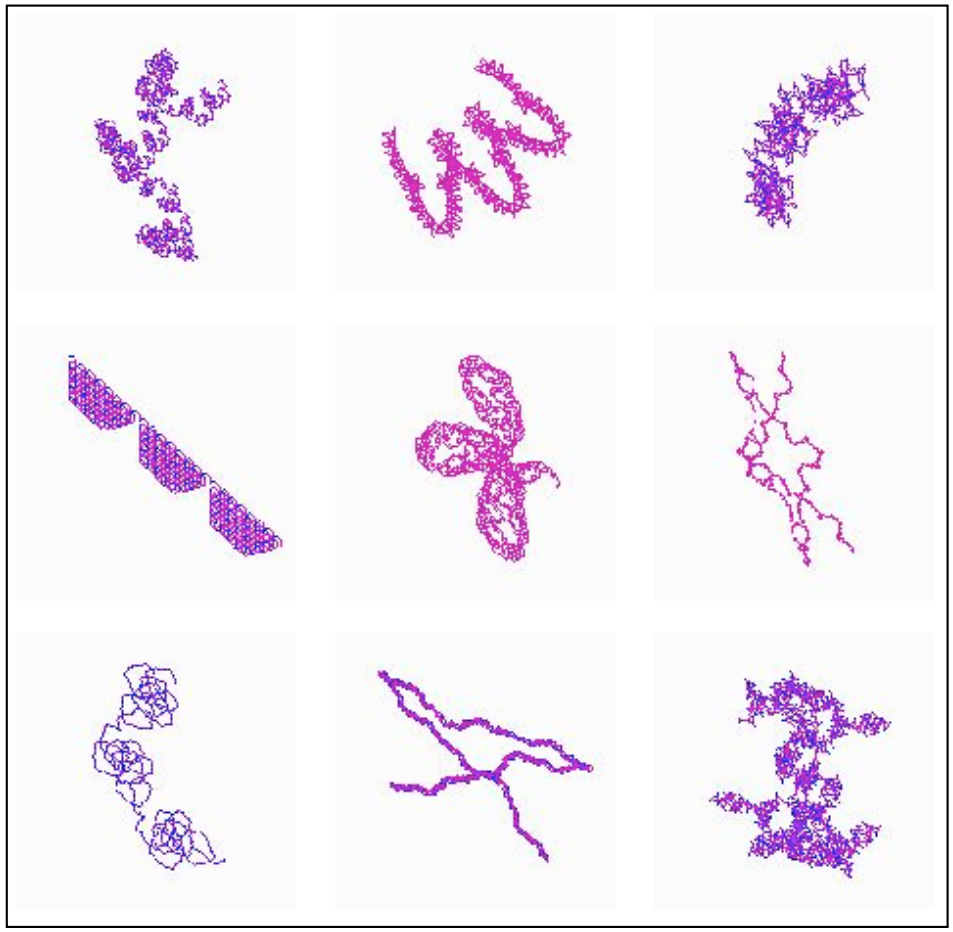
Gallery of Evolved Fractals



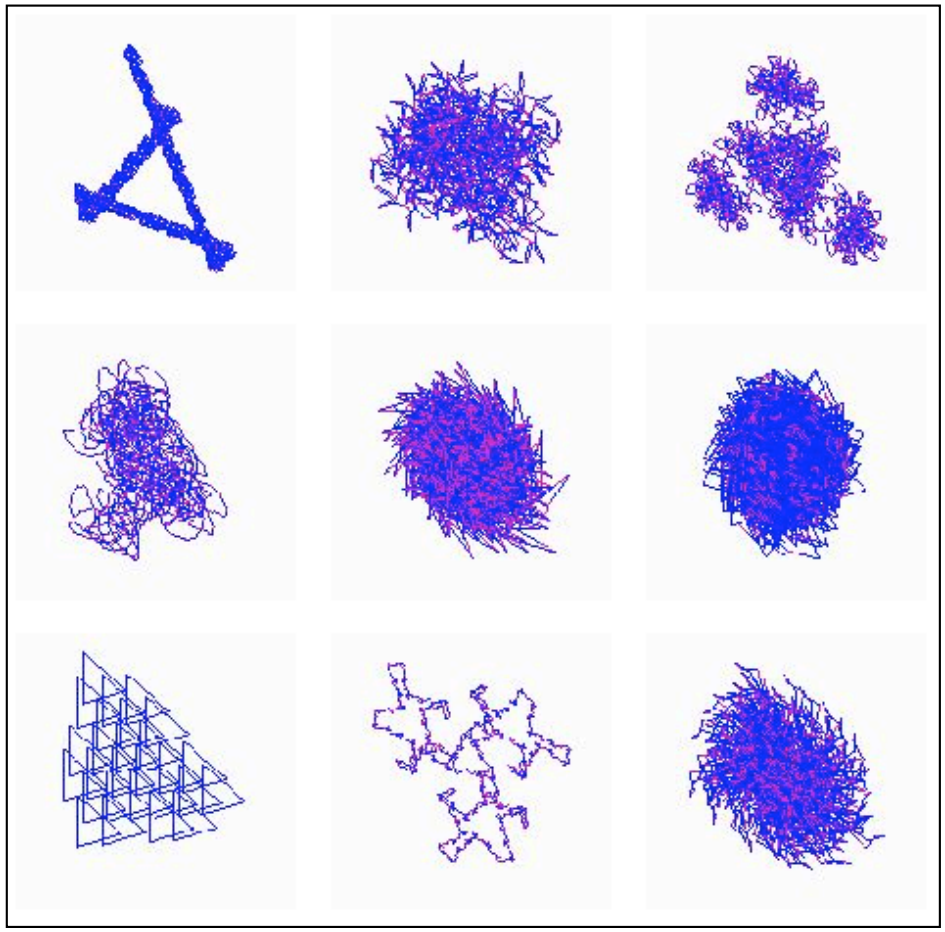
Gallery of Evolved Fractals



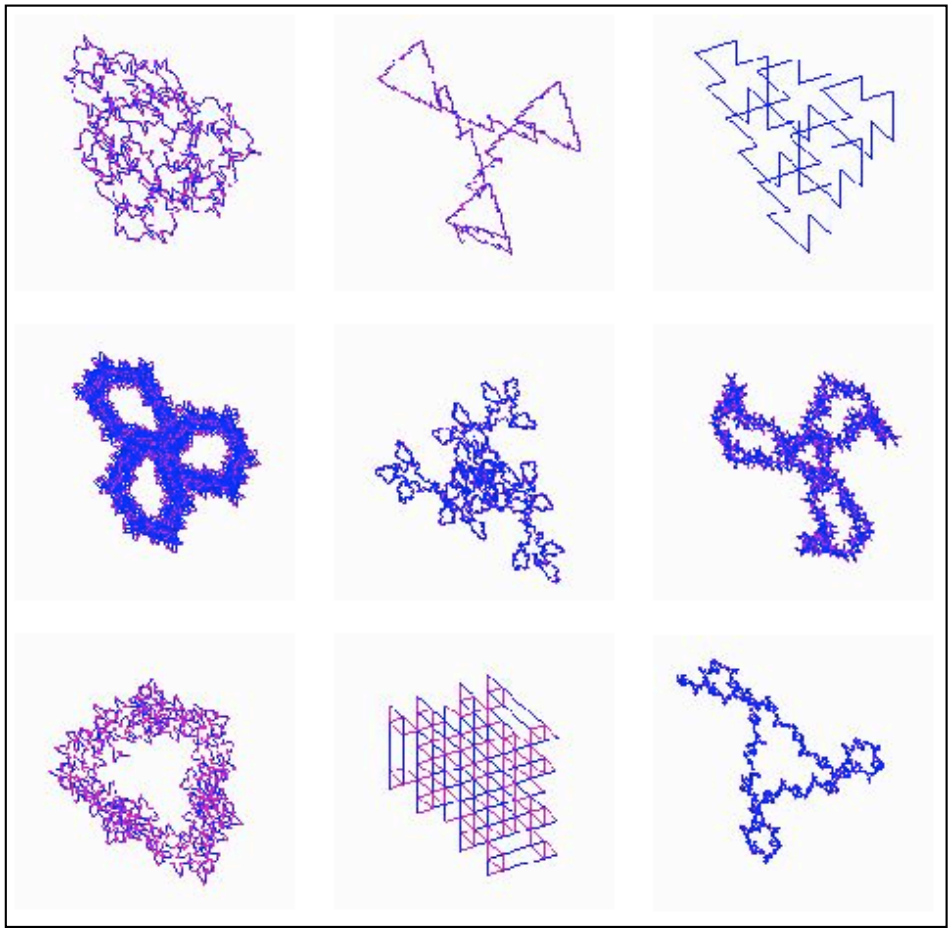
Gallery of Evolved Fractals



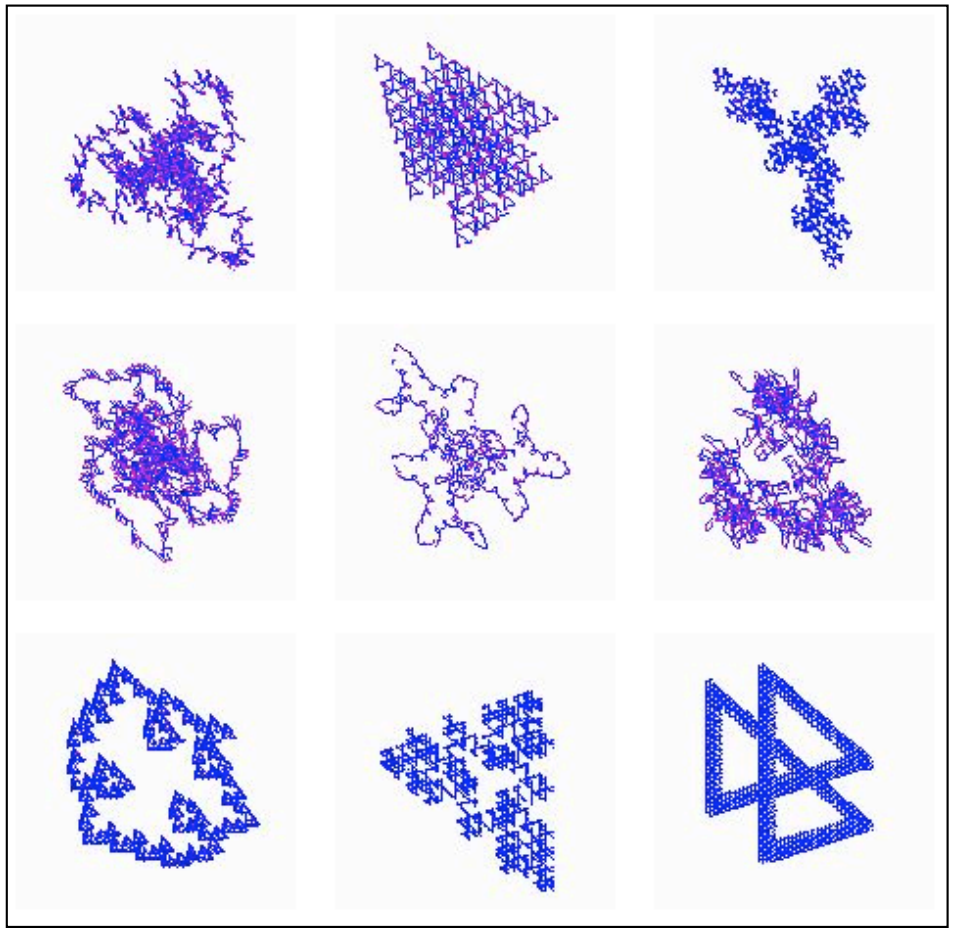
Gallery of Evolved Fractals



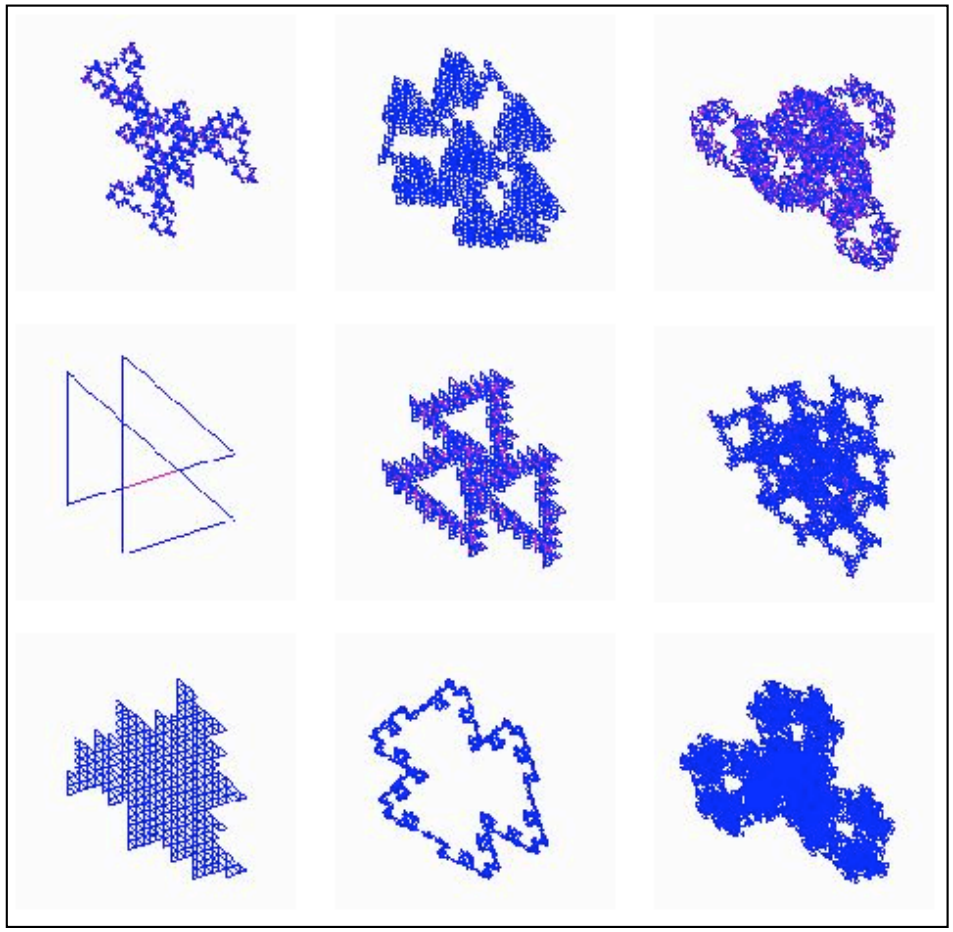
Gallery of Evolved Fractals



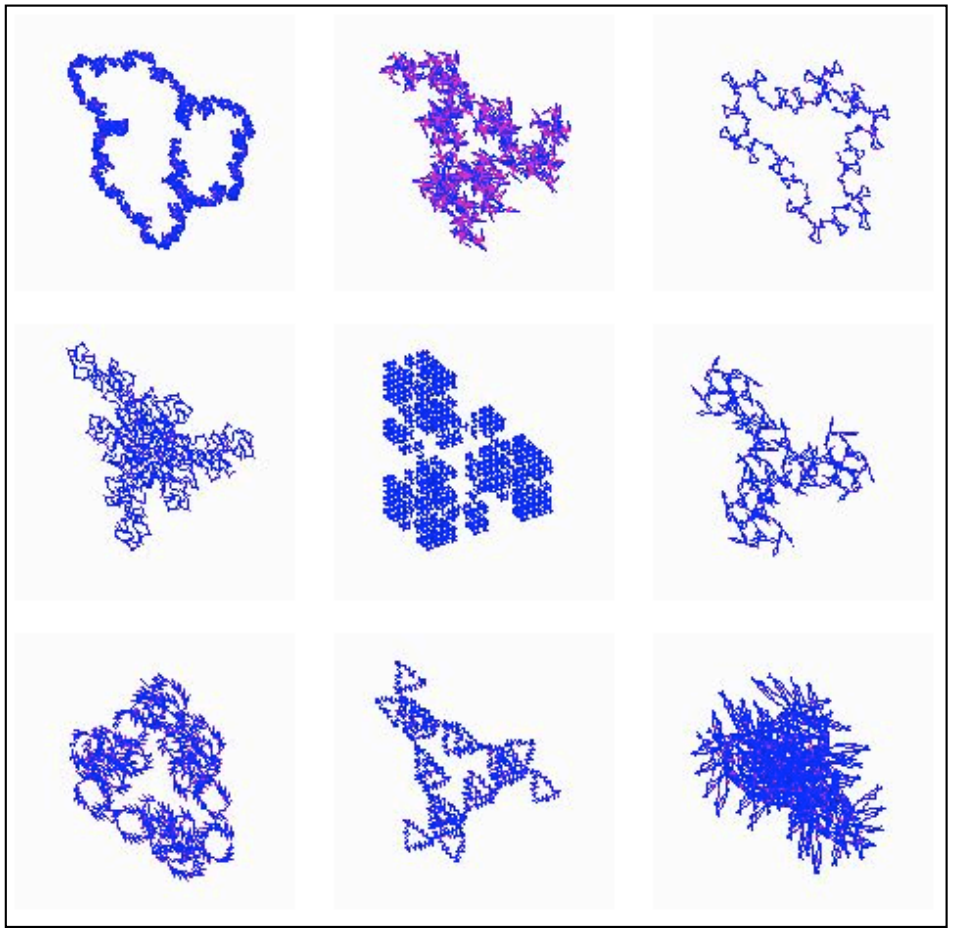
Gallery of Evolved Fractals



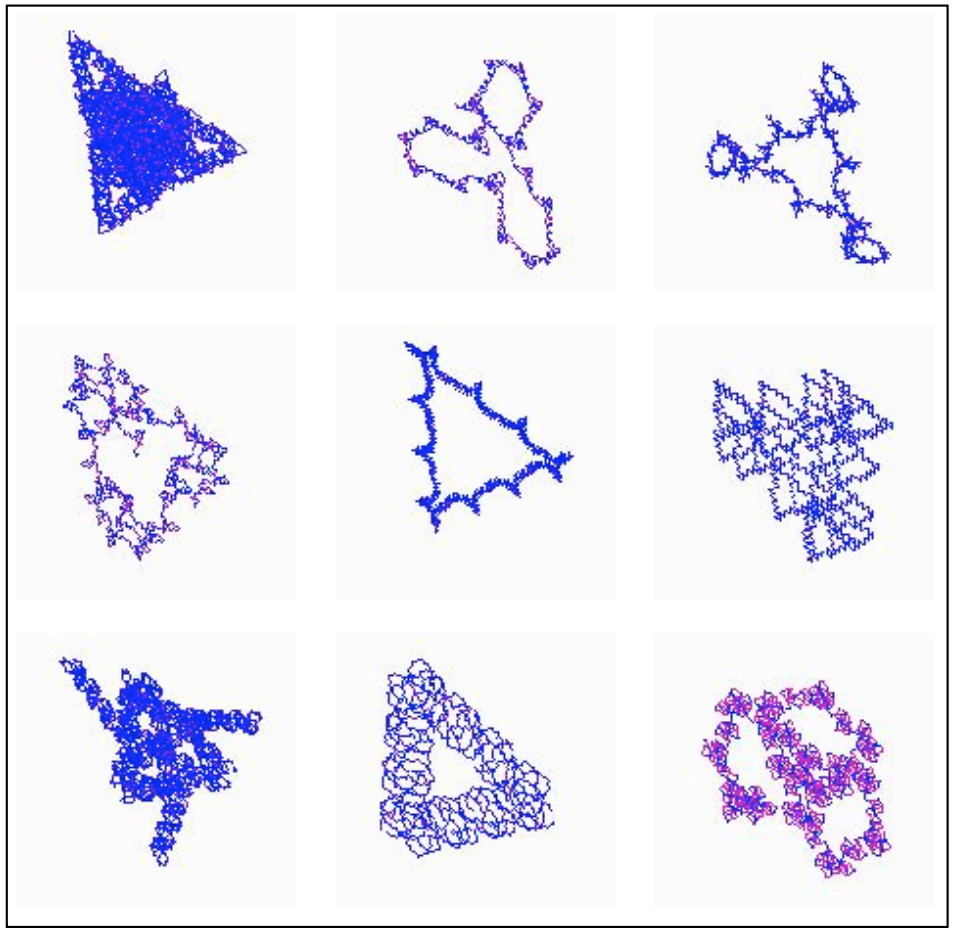
Gallery of Evolved Fractals



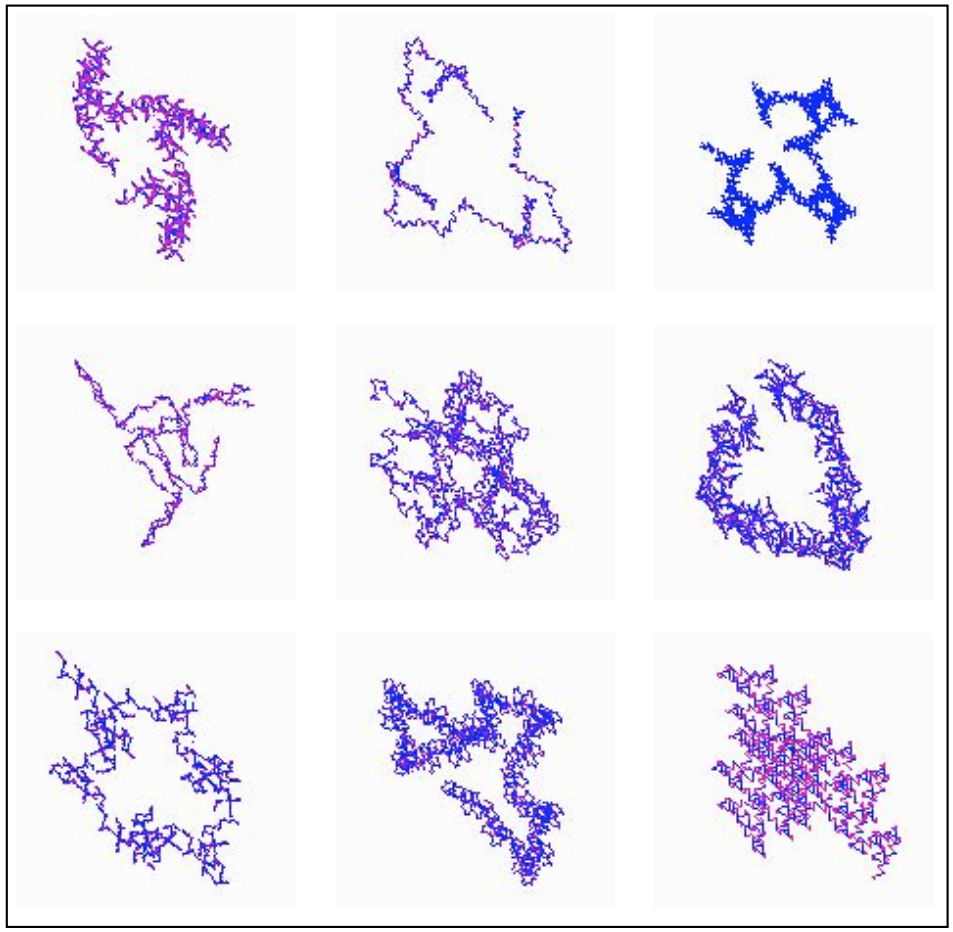
Gallery of Evolved Fractals



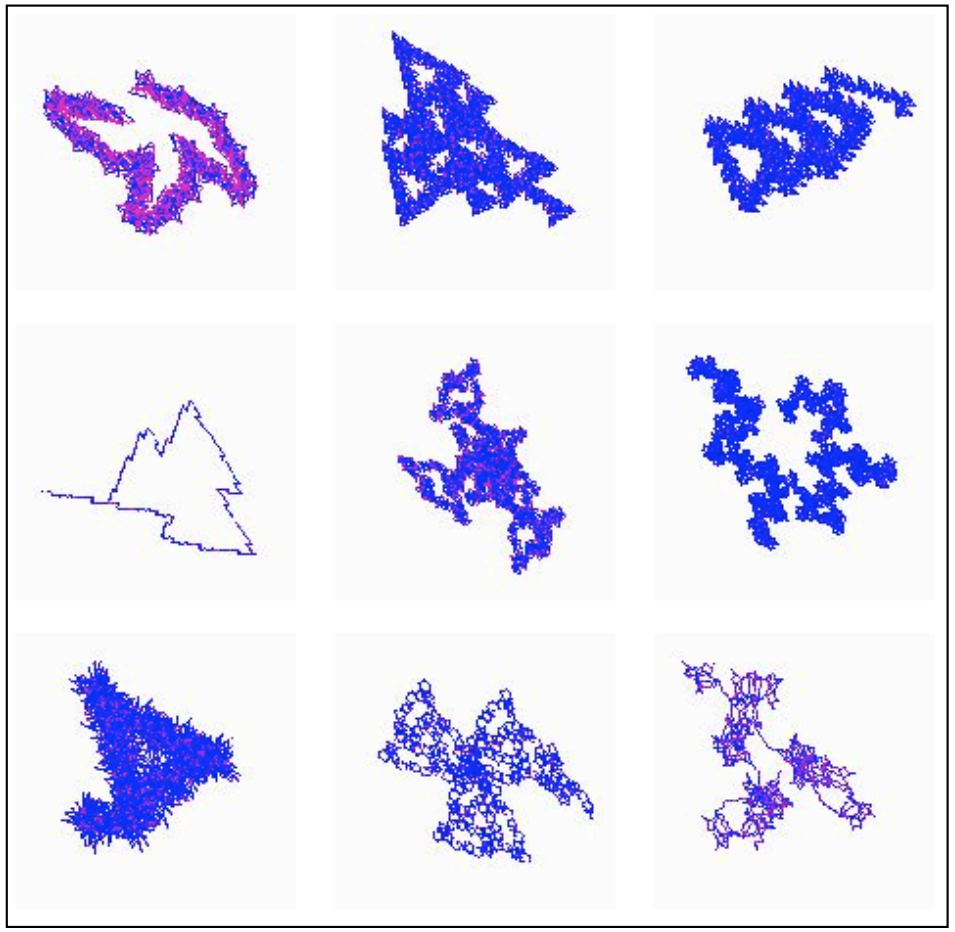
Gallery of Evolved Fractals



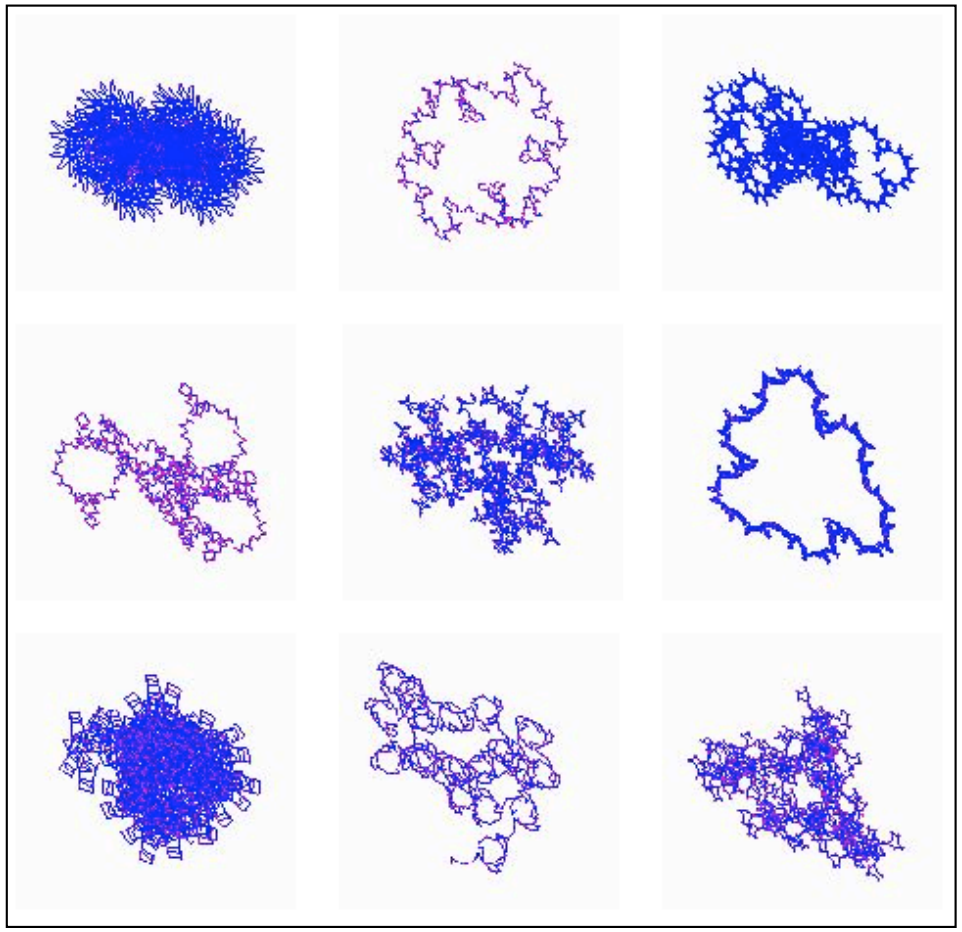
Gallery of Evolved Fractals



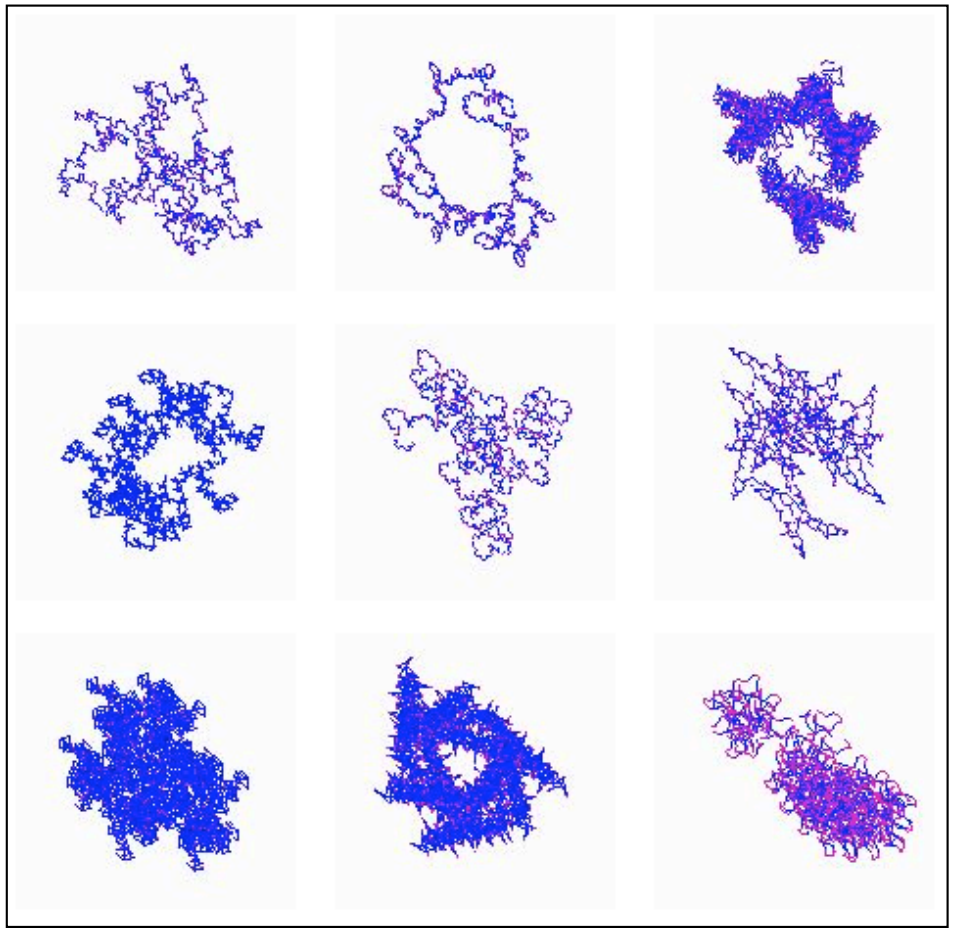
Gallery of Evolved Fractals



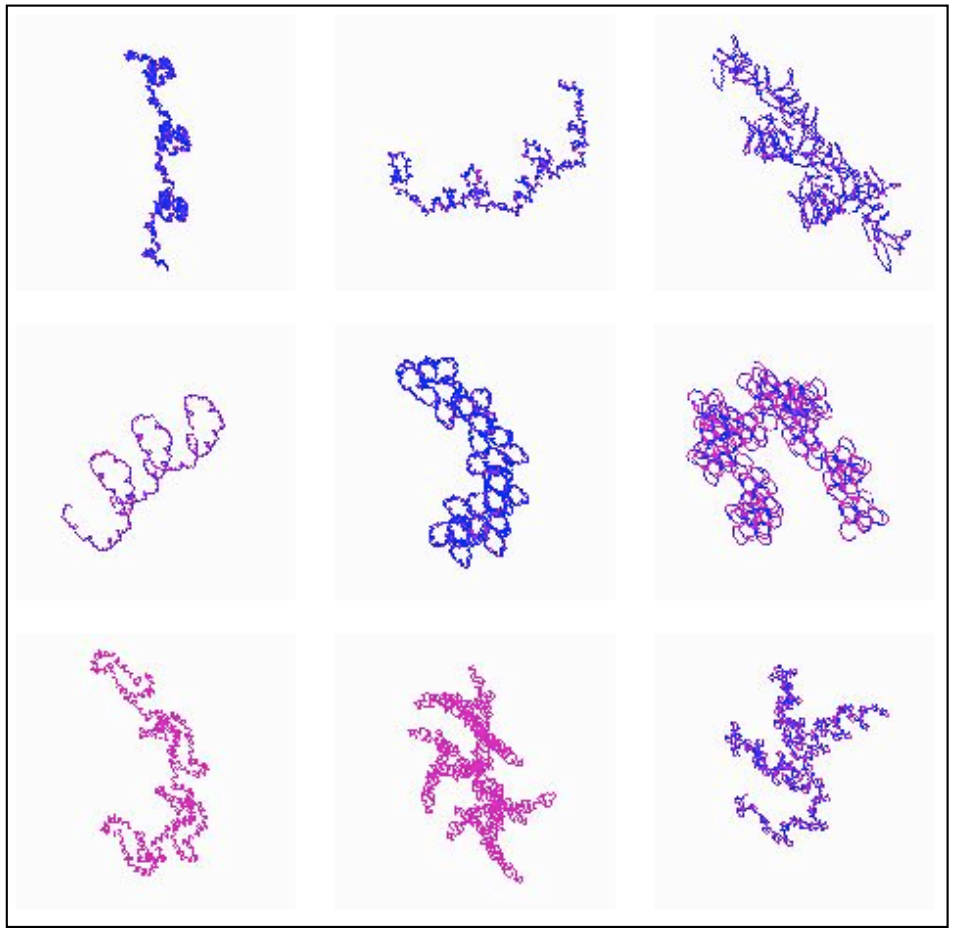
Gallery of Evolved Fractals



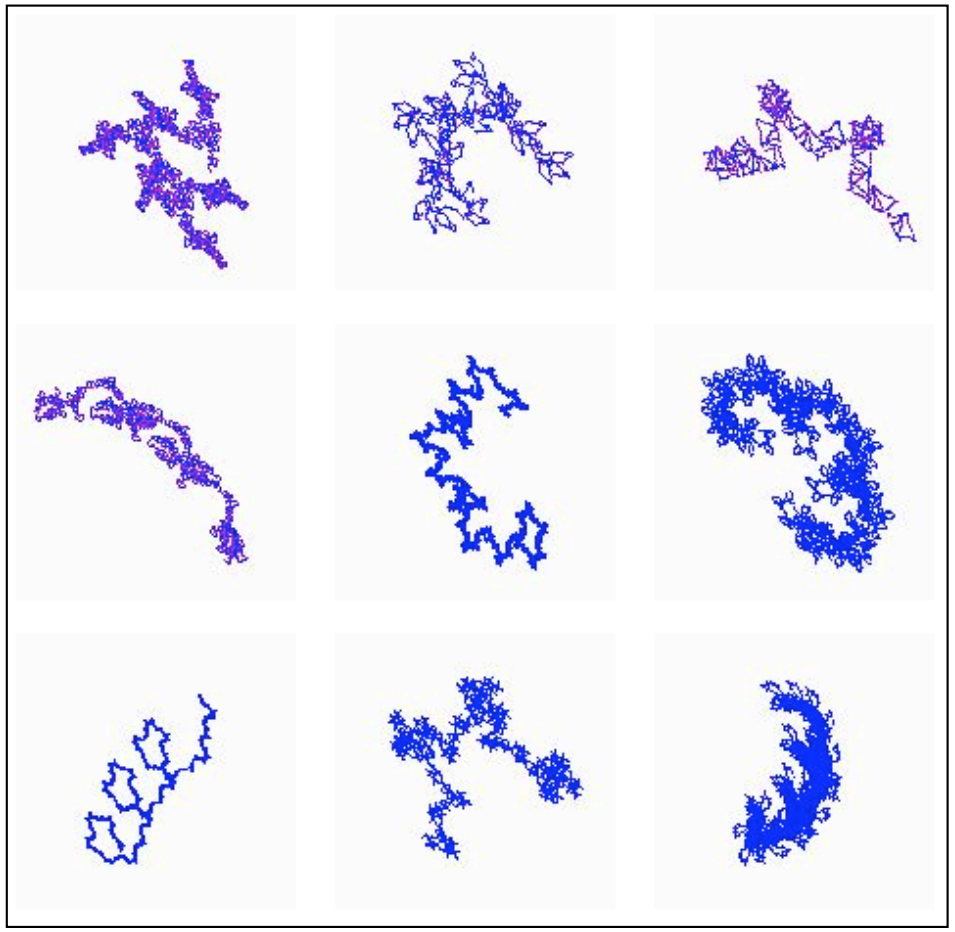
Gallery of Evolved Fractals



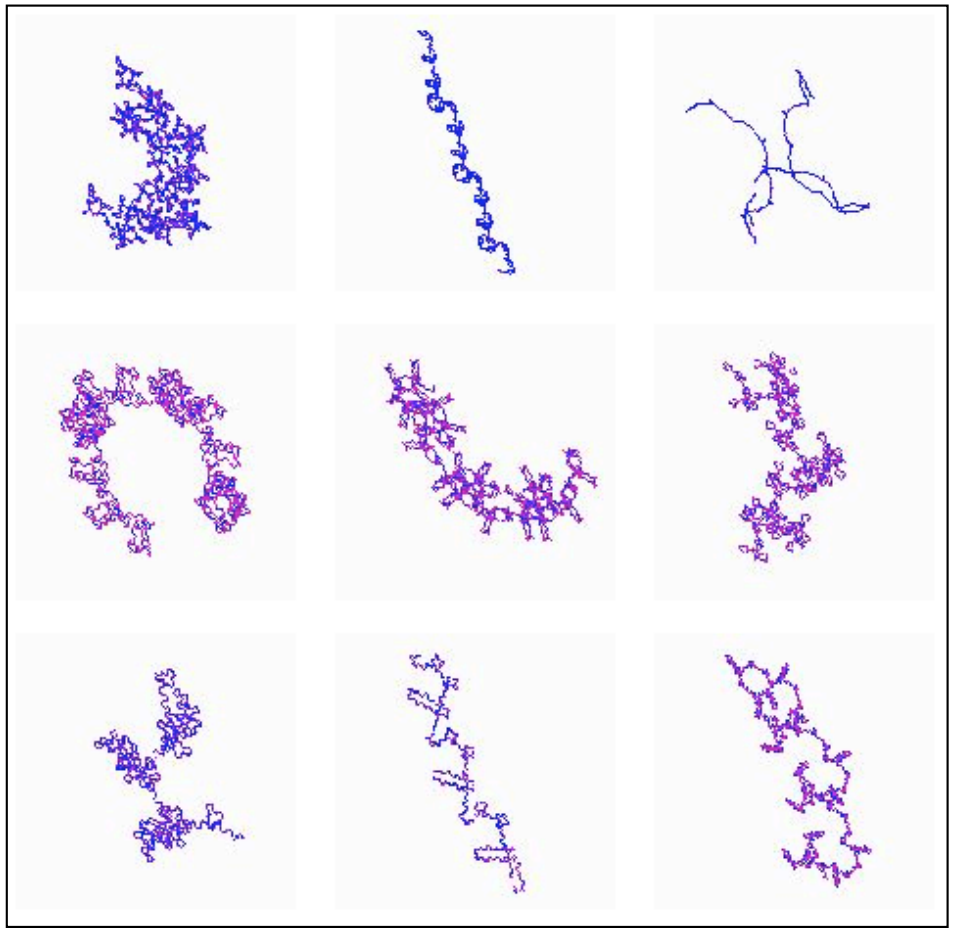
Gallery of Evolved Fractals



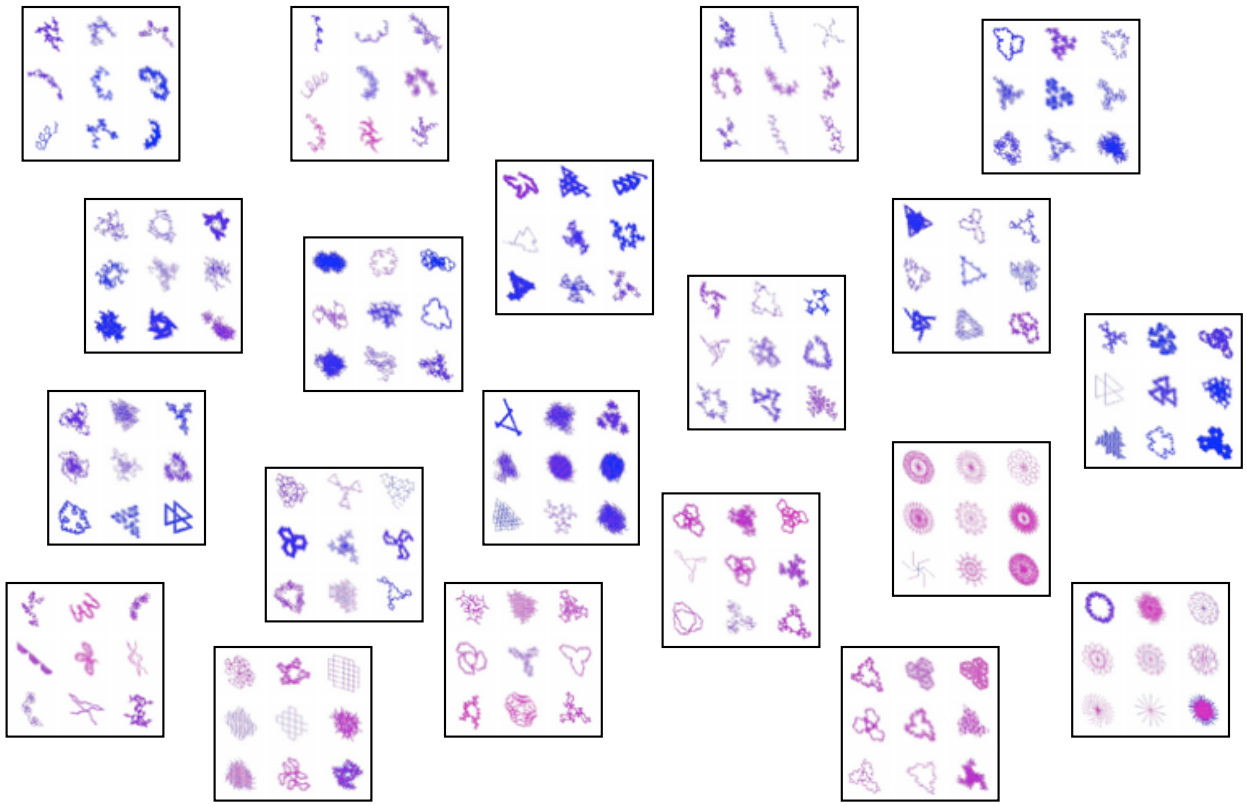
Gallery of Evolved Fractals



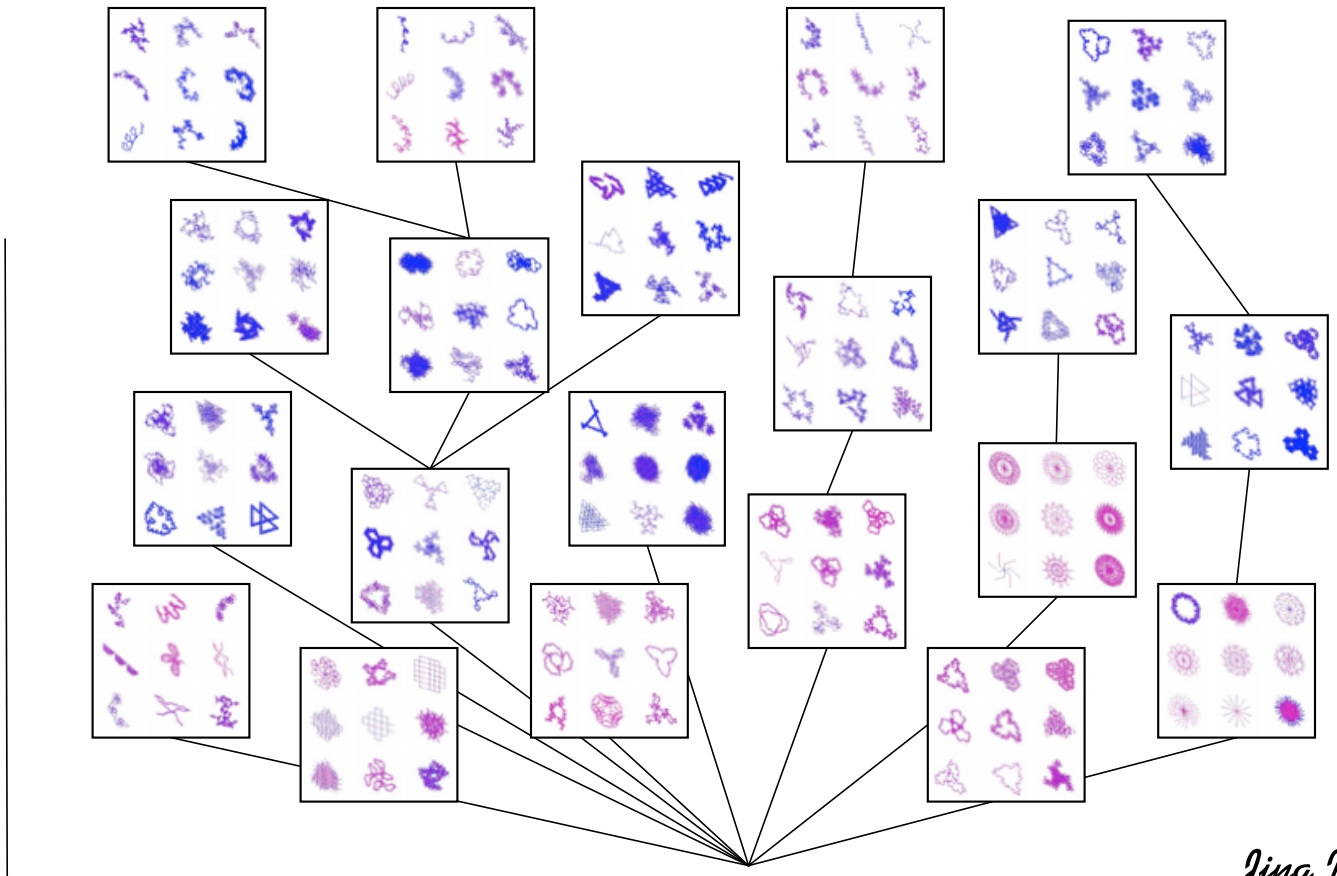
Gallery of Evolved Fractals



Evolutionary Exploration of Fractal Spaces



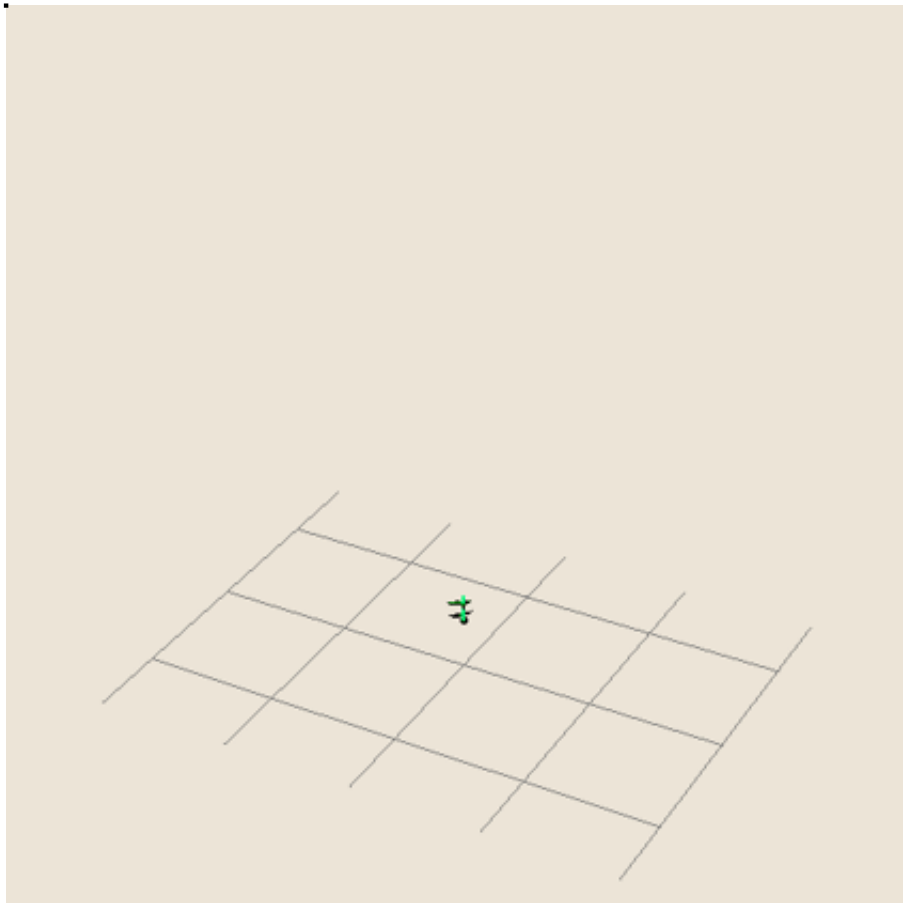
Evolutionary Exploration of Fractal Spaces



Jing Yu

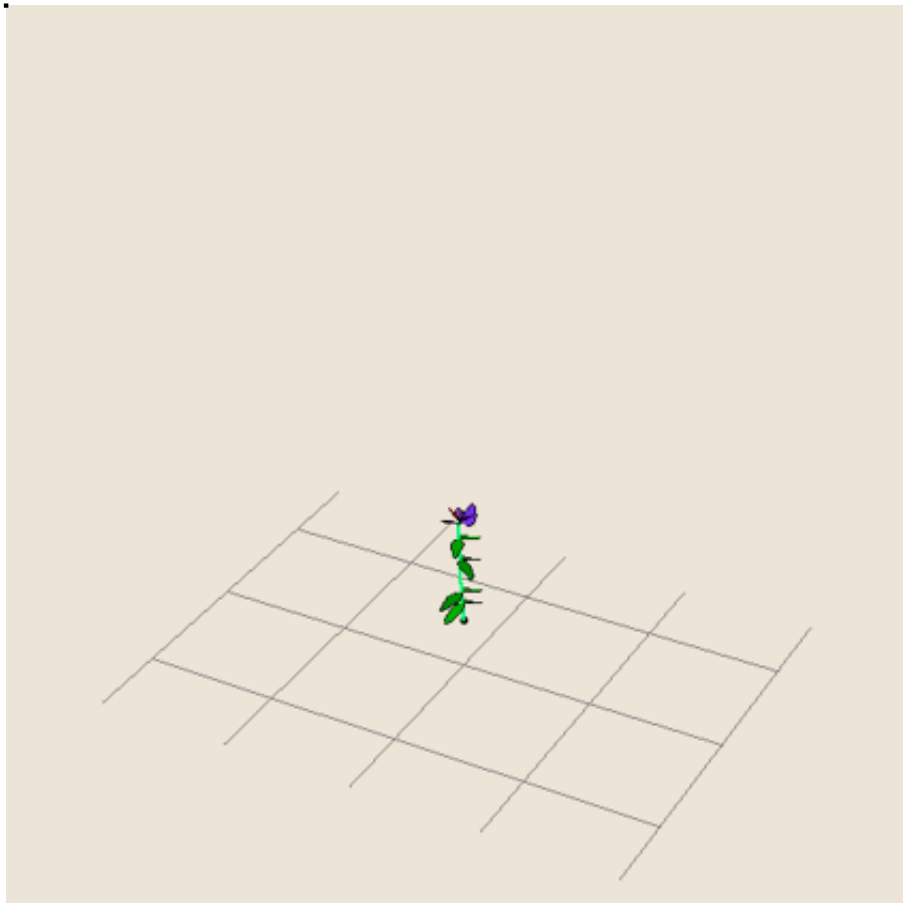
Evolutionary Inference of L-Systems

Developmental Programs



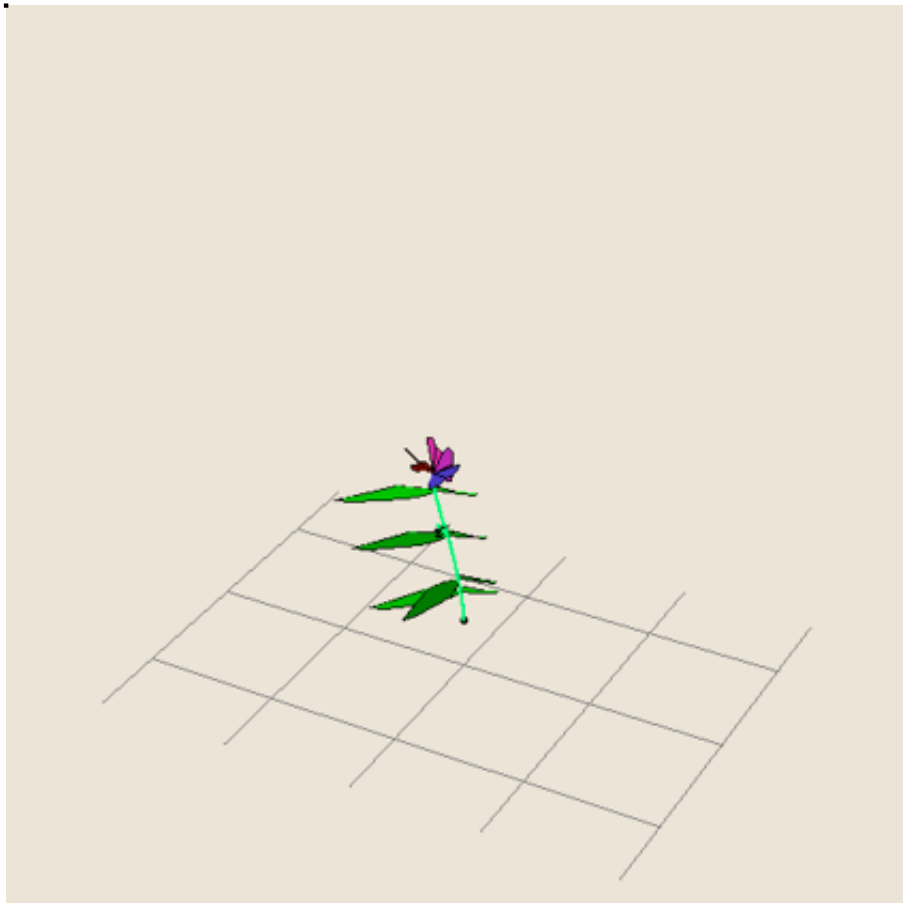
Iteration 1

Developmental Programs



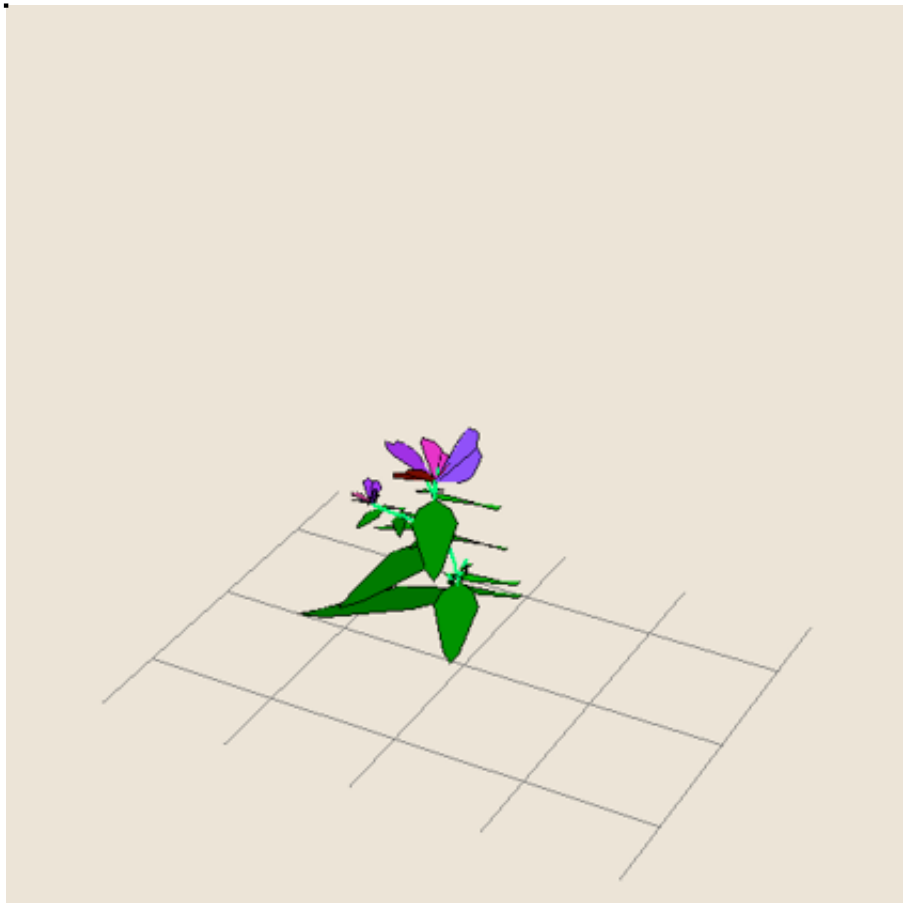
Iteration 2

Developmental Programs



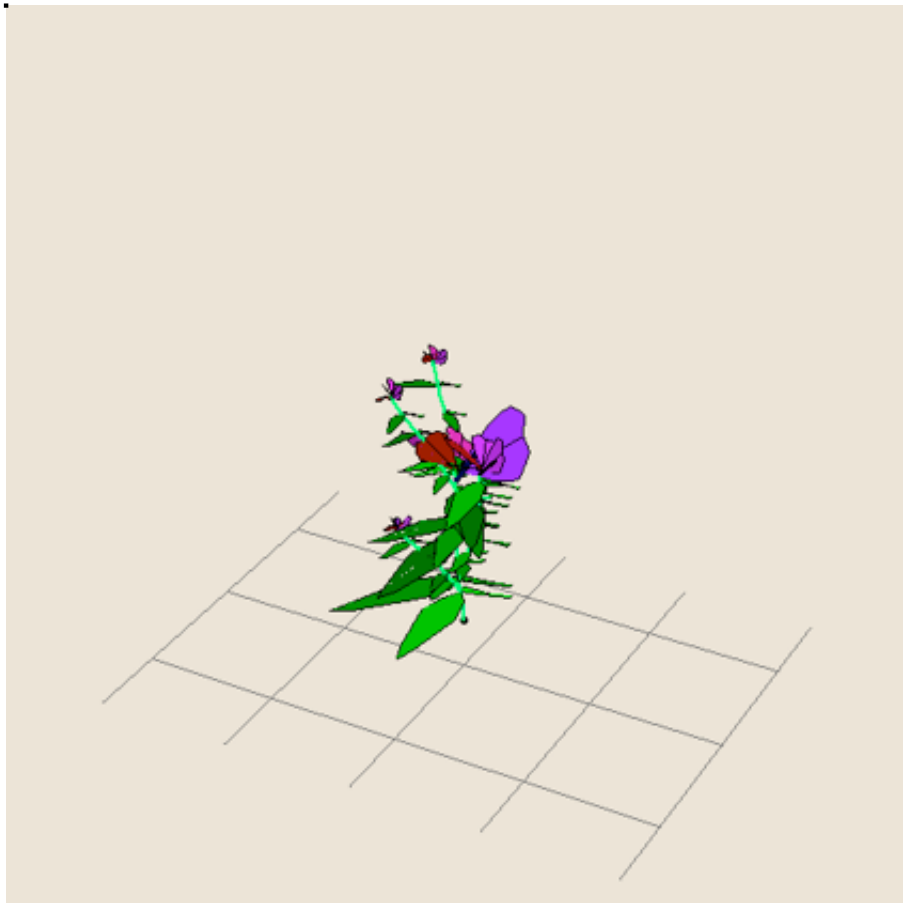
Iteration 3

Developmental Programs



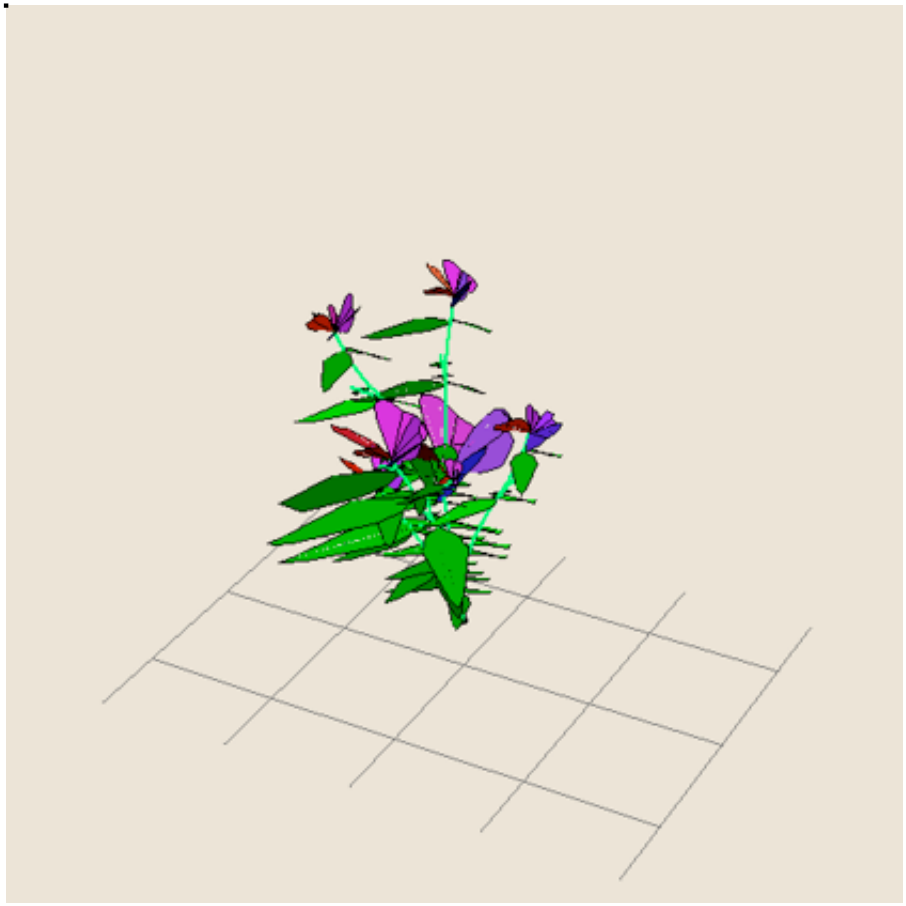
Iteration 4

Developmental Programs



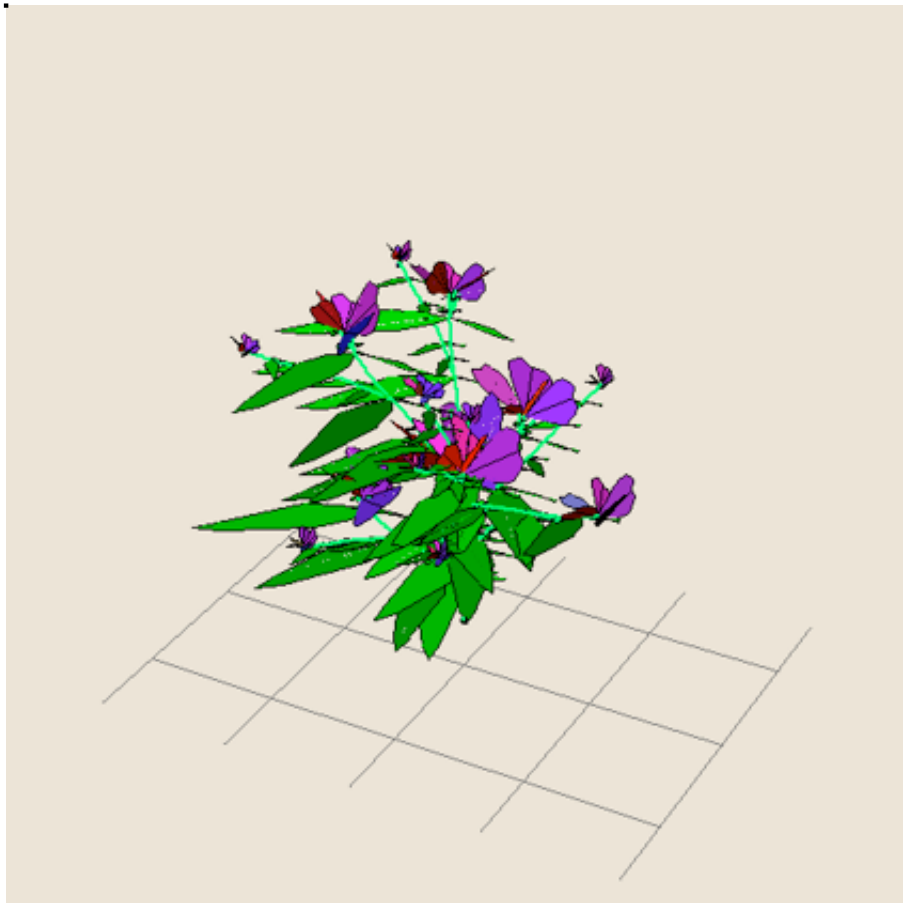
Iteration 5

Developmental Programs



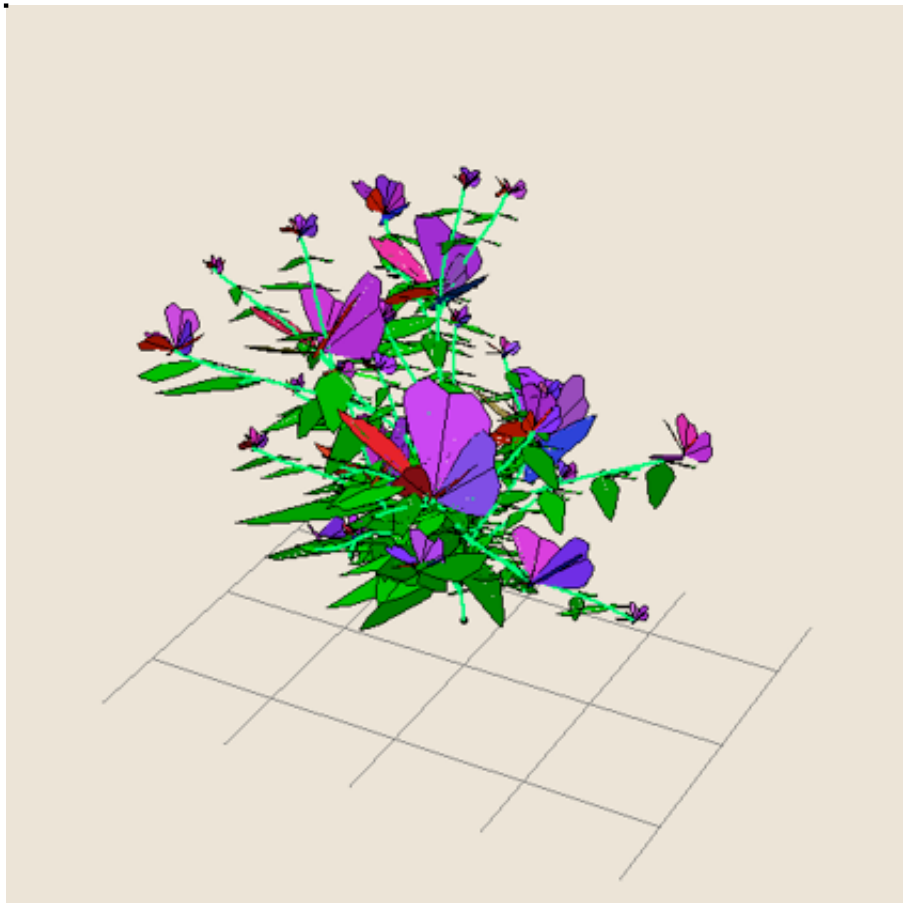
Iteration 6

Developmental Programs



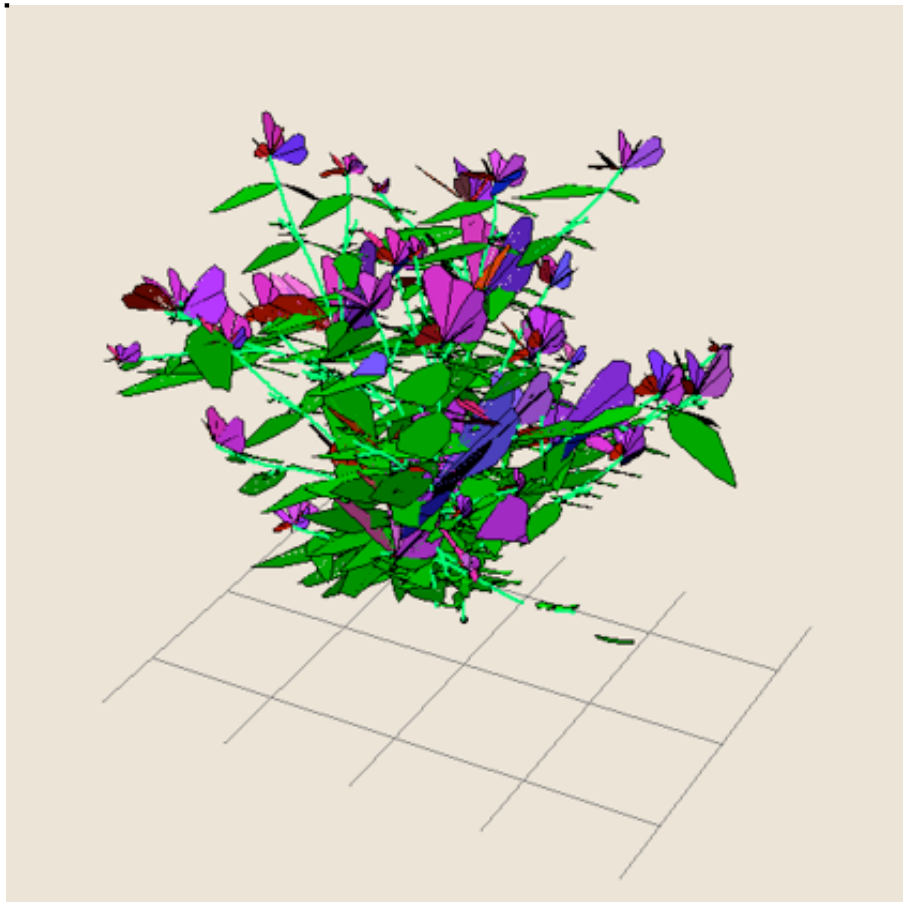
Iteration 7

Developmental Programs



Iteration 8

Developmental Programs



Iteration 9

ArtFlowers Encoded by L-systems

- Axiom: `sprout(4)`
- Sprout:

```
p1: sprout(4) -> f stalk(2) [ pd(60) leaf(0) ] pu(20) [ pu(25) sprout(0) ]
                [ pd(60) leaf(0) ] pd(29) [ pu(25) sprout(2) ] f stalk(1) bloom(0)
```

- Sprout riping:

```
p2: sprout(t<4) -> sprout(t+1)
```

- Stalk elongation:

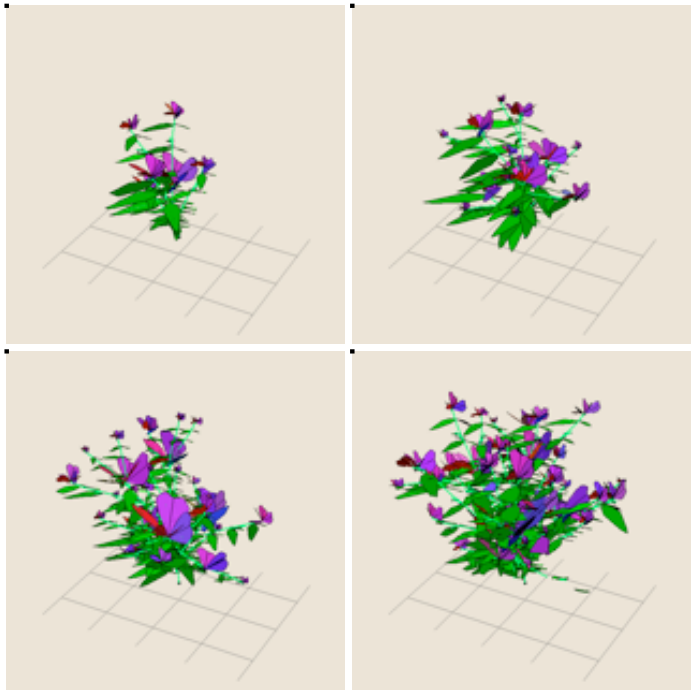
```
p3: stalk(t>0)  -> f f stalk(t-1)
```

- Bloom growth and decay:

```
p8: bloom(t)    -> bloom(t+1)
p9: bloom(7)    -> bloom(1)
```

- Changing leaf sizes:

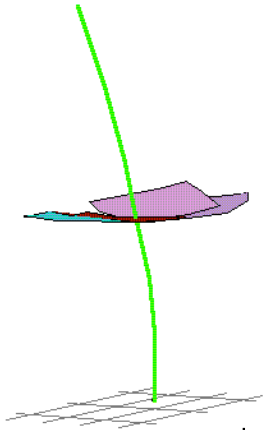
```
p4: leaf(t)     -> leaf(t+1.5)
p5: leaf(t>7)   -> Leaf(7)
p6: Leaf(t)     -> Leaf(t-1.5)
p7: Leaf(t<2)   -> leaf(0)
```



Lychnis coronaria artificialis

- Axiom: `sprout(a) ; a = 3`
- Sprout with emerging stalk, blooms and leaves

```
p1: sprout(a) -> f stalk(2) [ pd(60) leaf(0) ]
                 pu(20) [ pu(25) sprout(0) ]
                 [ pd(60) leaf(0) ] pd(29)
                 [ pu(25) sprout(2) ]
                 f stalk(1) bloom(0)
```

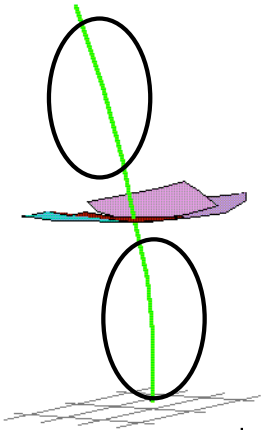


Iteration 1

Lychnis coronaria artificialis

- Axiom: `sprout(a)`
- Sprout with emerging stalk, blooms and leaves

```
p1: sprout(a) -> [ stalk(2) ] pd(60) leaf(0) ]
                 pu(20) [ pu(25) sprout(0) ]
                 [ pd(60) leaf(0) ] pd(29)
                 [ pu(25) sprout(2) ]
                 [ stalk(1) bloom(0)
```

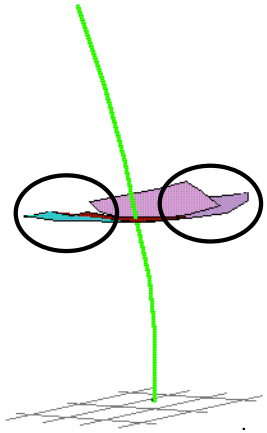


Iteration 1

Lychnis coronaria artificialis

- Axiom: `sprout(a)`
- Sprout with emerging stalk, blooms and leaves

```
p1: sprout(a) -> f stalk(2) [ pd(60) leaf(0) ]
                 pu(20) [ pu(25) sprout(0) ]
                 [ pd(60) leaf(0) ] pd(29)
                 [ pu(25) sprout(2) ]
                 f stalk(1) bloom(0)
```

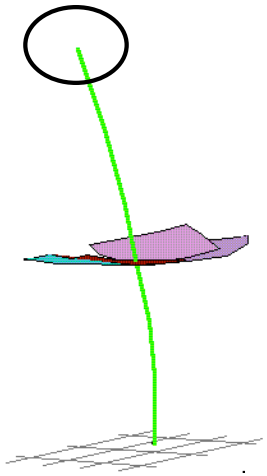


Iteration 1

Lychnis coronaria artificialis

- Axiom: `sprout(a)`
- Sprout with emerging stalk, blooms and leaves

```
p1: sprout(a) -> f stalk(2) [ pd(60) leaf(0) ]
                 pu(20) [ pu(25) sprout(0) ]
                 [ pd(60) leaf(0) ] pd(29)
                 [ pu(25) sprout(2) ]
                 f stalk(10) bloom(0)
```



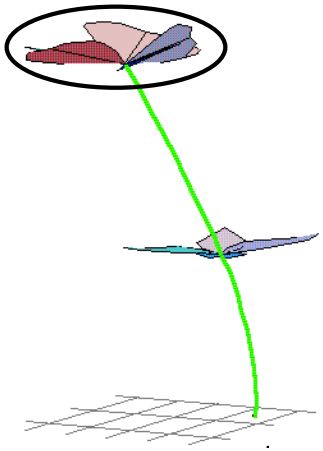
Iteration 1

Lychnis coronaria artificialis

Axiom: `sprout(a)`

Sprout with emerging stalk, blooms and leaves

```
p1: sprout(a)  ->      f stalk(2) [ pd(60) leaf(0) ]
                       pu(20) [ pu(25) sprout(0) ]
                       [ pd(60) leaf(0) ] pd(29)
                       [ pu(25) sprout(2) ]
                       f stalk(1) bloom(1)
```



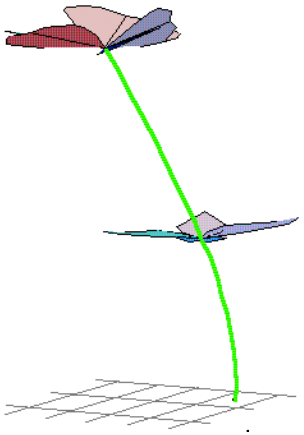
Iteration 2

Lychnis coronaria artificialis

Axiom: `sprout(a)`

Sprout with emerging stalk, blooms and leaves

```
p1: sprout(a)  ->      f stalk(2) [ pd(60) leaf(0) ]
                       pu(20) [ pu(25) sprout(0) ]
                       [ pd(60) leaf(0) ] pd(29)
                       [ pu(25) sprout(2) ]
                       f stalk(1) bloom(1)
```



Iteration 2

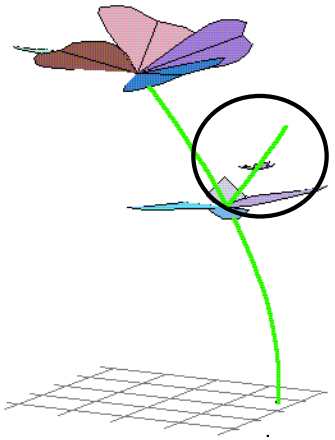
Lychnis coronaria artificialis

Axiom: `sprout(a)`

Sprout with emerging stalk, blooms and leaves

p_1: `sprout(a)` ->

```
f stalk(2) [ pd(60) leaf(0) ]
pu(20) [ pu(25) sprout(0) ]
[ pd(60) leaf(0) ] pd(29)
[ pu(25) f stalk(2) [ pd(60) leaf(0)]
pu(20) [ pu(25) sprout(0) ]
[ pd(60) leaf(0) ] pd(29)
[ pu(25) sprout(2) ] f stalk(1) bloom(0)]
f stalk(1) bloom(1)
```



Iteration 3

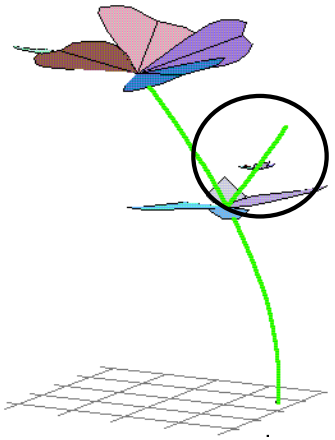
Lychnis coronaria artificialis

Axiom: `sprout(a)`

Sprout with emerging stalk, blooms and leaves

p_1: `sprout(a)` ->

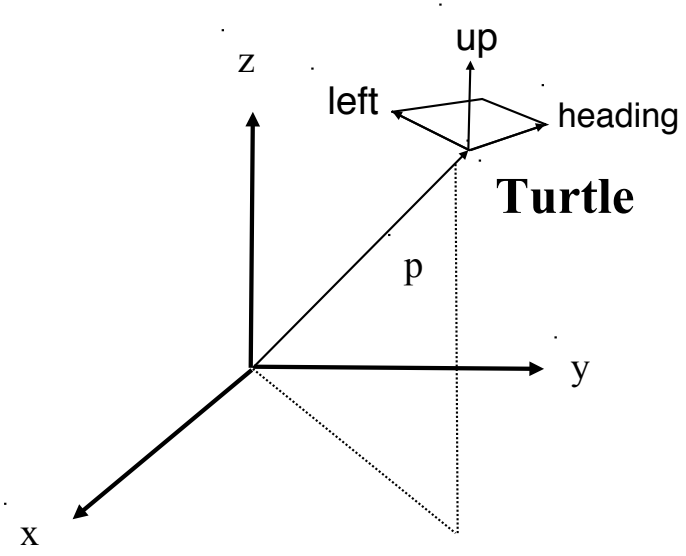
```
f stalk(2) [ pd(60) leaf(0) ]
pu(20) [ pu(25) sprout(0) ]
[ pd(60) leaf(0) ] pd(29)
[ pu(25) f stalk(2) [ pd(60) leaf(0)]
pu(20) [ pu(25) sprout(0) ]
[ pd(60) leaf(0) ] pd(29)
[ pu(25) sprout(2) ] f stalk(1) bloom(0)]
f stalk(1) bloom(1)
```



Iteration 3

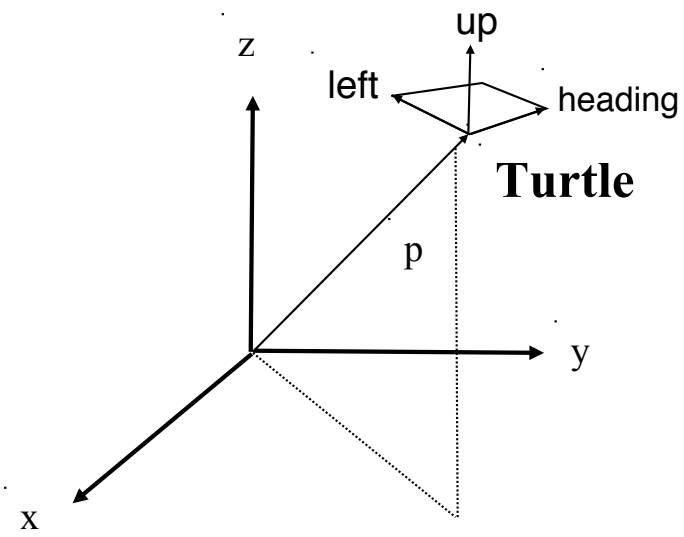
Turtle Interpretation

Position and Orientation

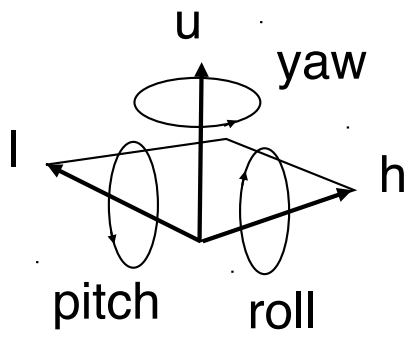


Turtle Interpretation

Position and Orientation



Changing Orientation



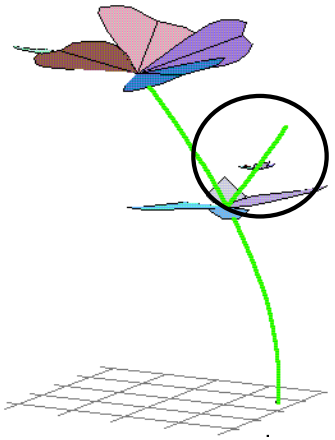
Lychnis coronaria artificialis

Axiom: `sprout(a)`

Sprout with emerging stalk, blooms and leaves

p_1: `sprout(a)` ->

```
f stalk(2) [ pd(60) leaf(0) ]
pu(20) [ pu(25) sprout(0) ]
[ pd(60) leaf(0) ] pd(29)
[ pu(25) f stalk(2) [ pd(60) leaf(0)]
pu(20) [ pu(25) sprout(0) ]
[ pd(60) leaf(0) ] pd(29)
[ pu(25) sprout(2) ] f stalk(1) bloom(0)]
f stalk(1) bloom(1)
```



Iteration 3

ArtFlowers Encoded by L-systems

- Axiom: `sprout(4)`
- Sprout:

```
p1: sprout(4) -> f stalk(2) [ pd(60) leaf(0) ] pu(20) [ pu(25) sprout(0) ]
                [ pd(60) leaf(0) ] pd(29) [ pu(25) sprout(2) ] f stalk(1) bloom(0)
```

- Sprout riping:

```
p2: sprout(t<4) -> sprout(t+1)
```

- Stalk elongation:

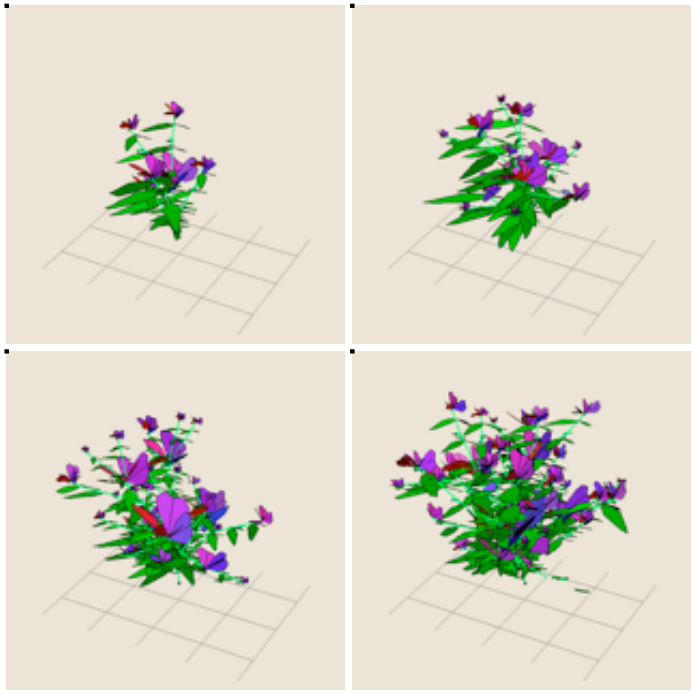
```
p3: stalk(t>0)  -> f f stalk(t-1)
```

- Bloom growth and decay:

```
p8: bloom(t)    -> bloom(t+1)
p9: bloom(7)    -> bloom(1)
```

- Changing leaf sizes:

```
p4: leaf(t)     -> leaf(t+1.5)
p5: leaf(t>7)   -> Leaf(7)
p6: Leaf(t)     -> Leaf(t-1.5)
p7: Leaf(t<2)   -> leaf(0)
```



Encoding L-Systems

- General form of an L-rule:

  Left < Predecessor > Right ⇨ Successor

Encoding L-Systems

- General form of an L-rule:

  Left < Predecessor > Right  Successor

- Encoding by a symbolic expression:

```
LRule[
  LEFT[Left], PRED[Predecessor],RIGHT[Right],
  SUCC[Successor]
]
```

Encoding L-Systems

- LSystem[

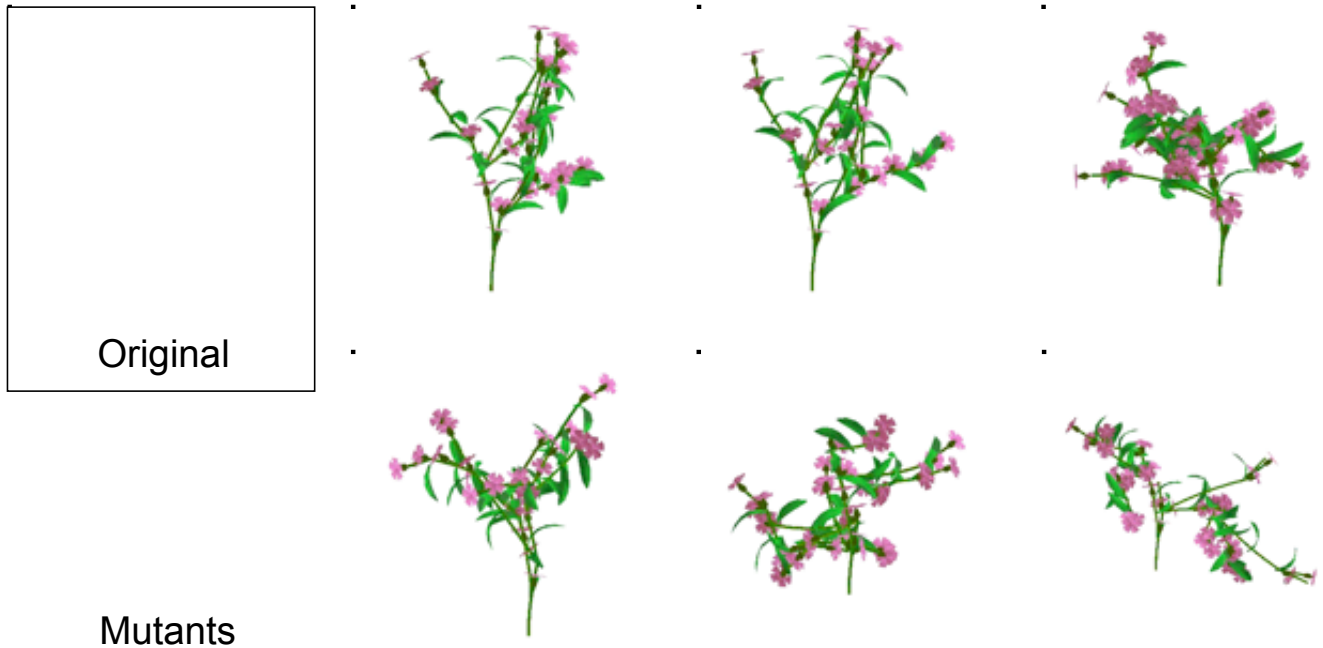
  Axiom[...],

  LRules[
    LRule[...],

    LRule[LEFT[*Left*],PRED[*Pred*],RIGHT[*Right*],
          SUCC[*Succ*]
    ],

    LRule[...], ...
  ]
]

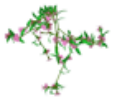
ArtFlowers Gallery



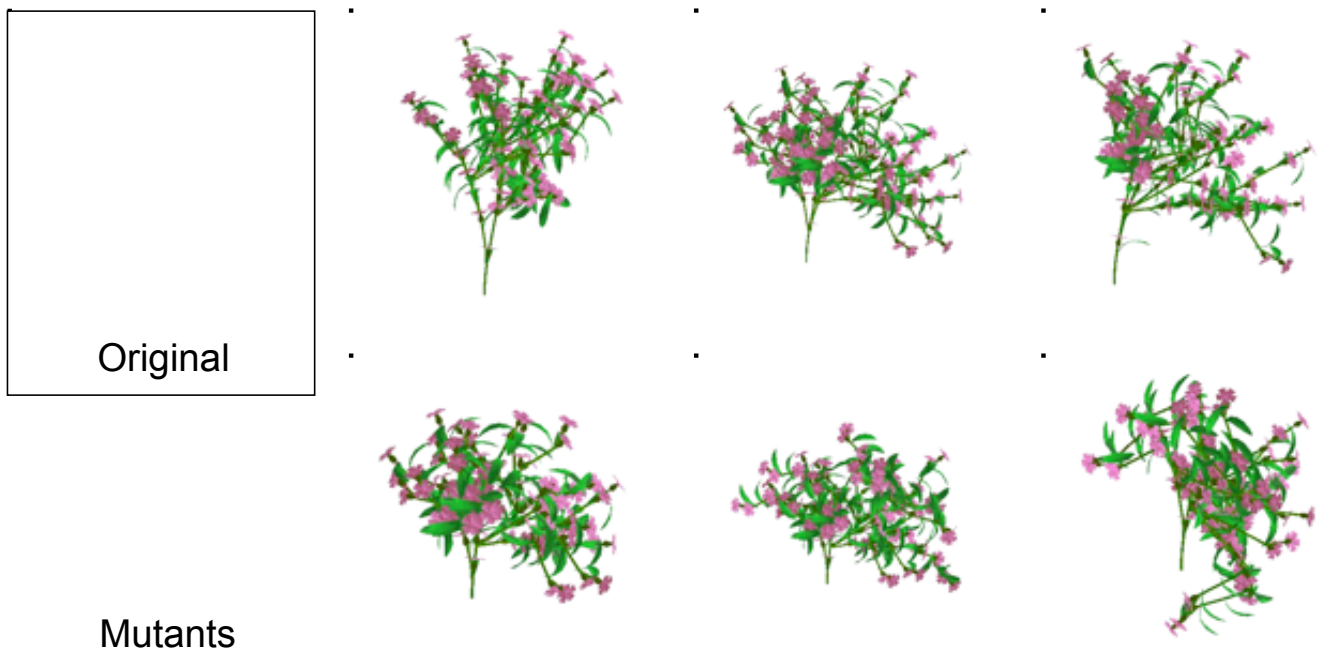
ArtFlowers Gallery

Original

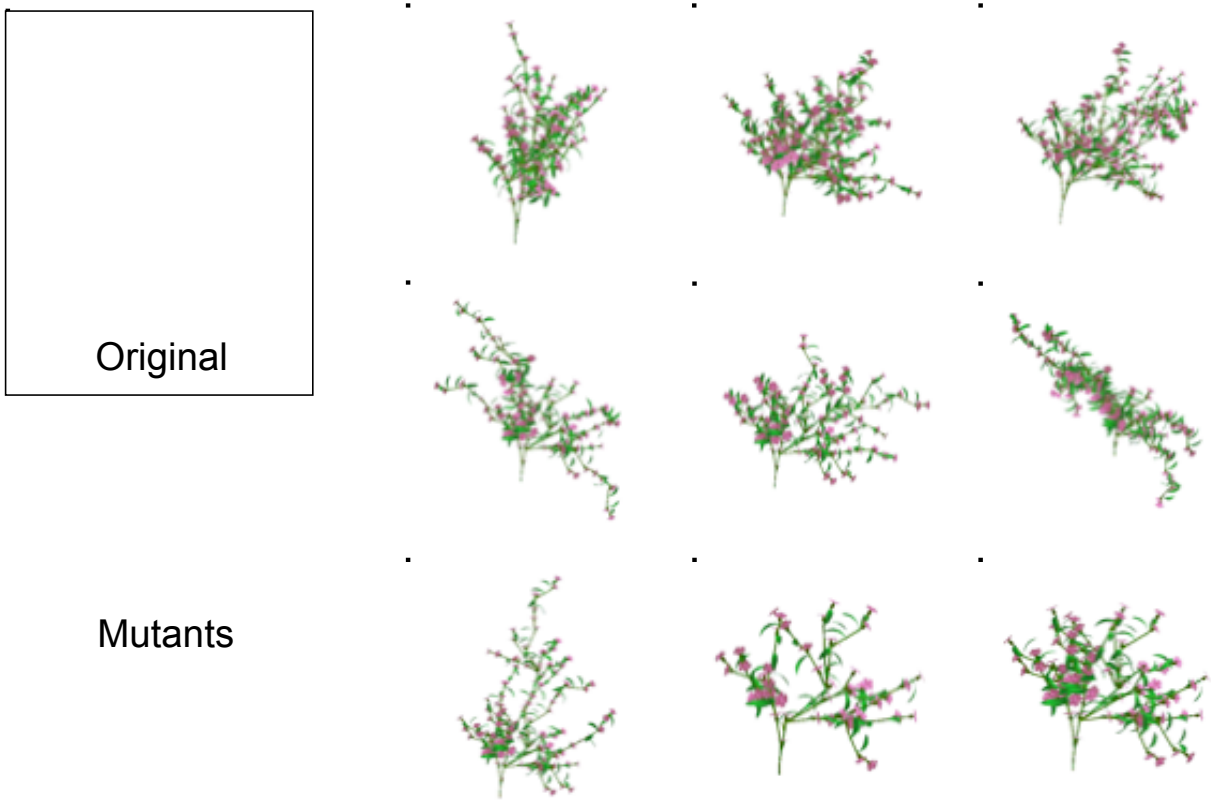
Mutants



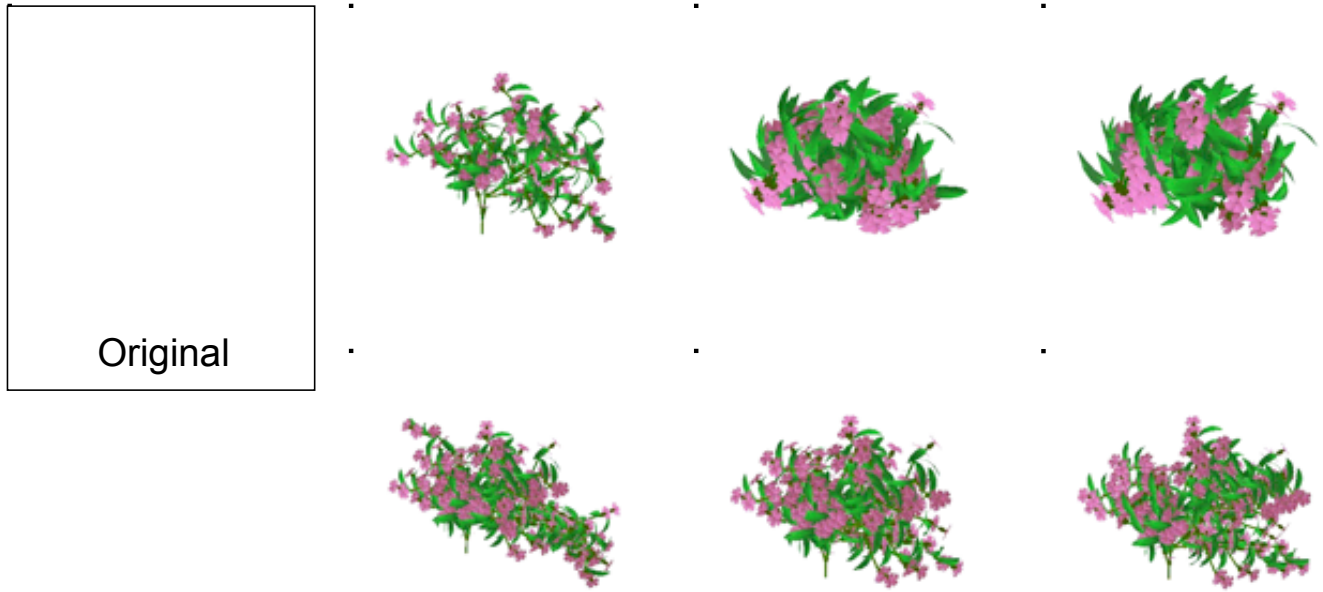
ArtFlowers Gallery



ArtFlowers Gallery



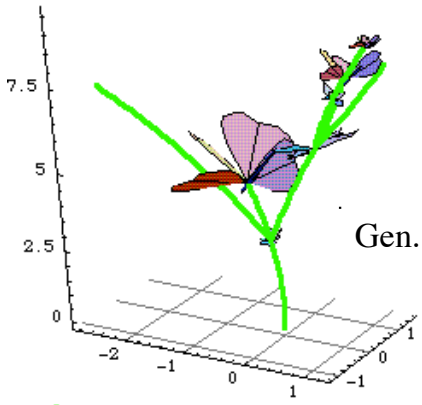
ArtFlowers Gallery



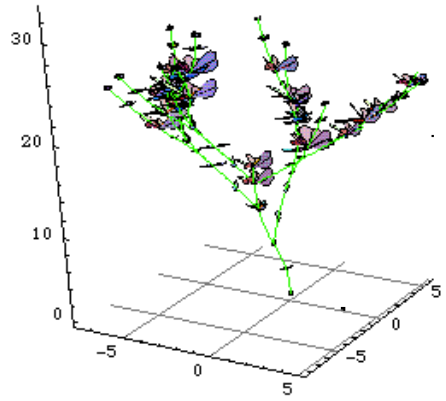
Visualizing Evolution and Ontogeny

Effects of Genetic Operators

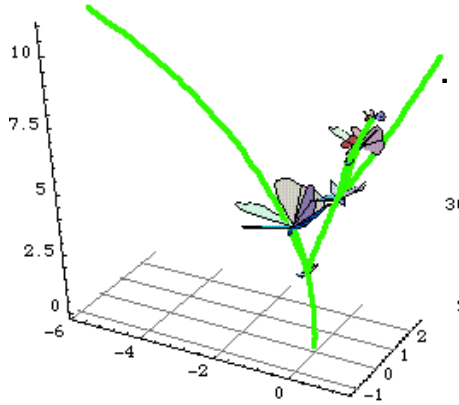
Effects of Genetic Operators



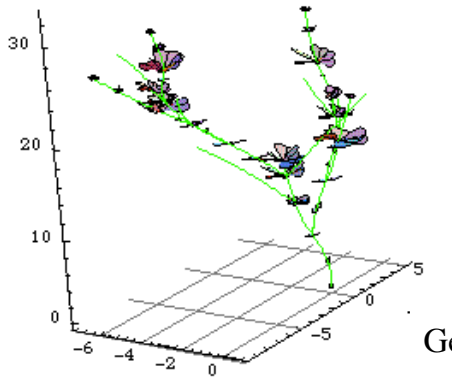
Gen. 1



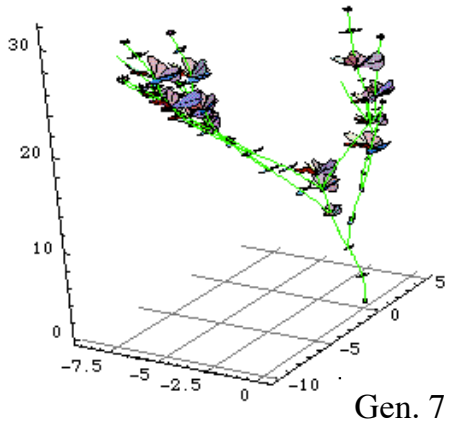
Gen. 8



Gen. 4



Gen. 6

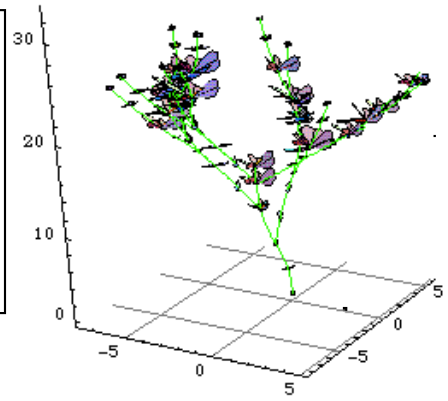


Gen. 7

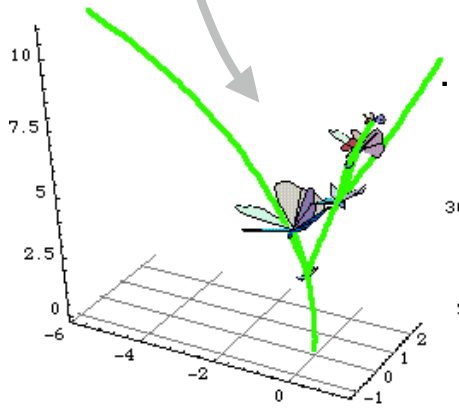
Effects of Genetic Operators

```
LSystem[AXIOM[a[4]],
 LRULES[
  LRule[LEFT[], PRED[a[4]], RIGHT[],
   SUCC[SEQ[SEQ[f], SEQ[f[2]],
    STACK[pd[60], [0]], SEQ[m[90]],
    STACK[pd[20], a[3]], SEQ[m[90]],
    SEQ[f[0], RR[17], [2]],
    STACK[pd[60], [0]], SEQ[m[90]],
    STACK[pd[20], a[2]], SEQ[f],
    SEQ[f[1], m[0]]],
  LRule[LEFT[], PRED[a[t_/;t<4]], RIGHT[], SUCC[a[1 + t]],
  --
  LRule[LEFT[], PRED[m[6]], RIGHT[], SUCC[m[1]]],
  LRule[LEFT[], PRED[a[2]], RIGHT[], SUCC[STACK[f[4], YR[34]]],
  LRule[LEFT[], PRED[a[0]], RIGHT[], SUCC[STACK[YR[18], YR[18]]]]
```

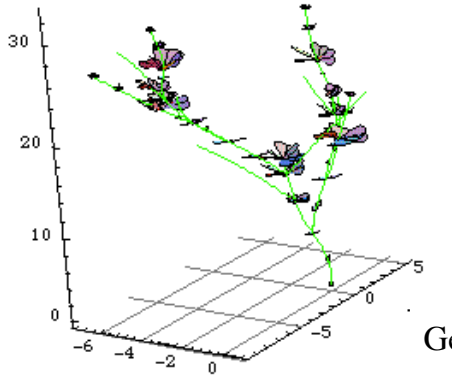
Duplication



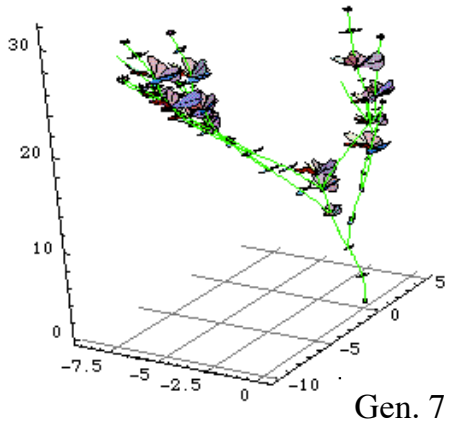
Gen. 8



Gen. 4



Gen. 6

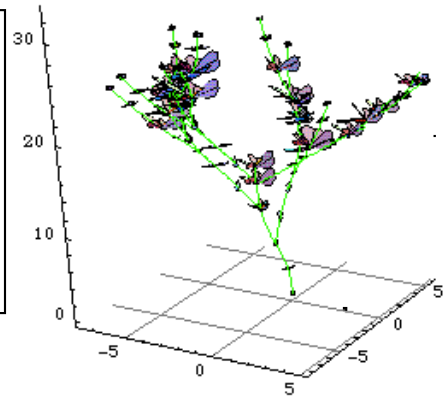


Gen. 7

Effects of Genetic Operators

```
LSystem[AXIOM[a[4]],
 LRULES{
  LRule[LEFT[], PRED[a[4]], RIGHT[],
   SUCC[SEQ[SEQ[f], SEQ[i[2],
    STACK[pd[60], l[0]], SEQ[rr[90]],
    STACK[pd[20], a[3]], SEQ[rr[90]],
    SEQ[i[0], RR[17], l[2],
    STACK[pd[60], l[0]], SEQ[rr[90]],
    STACK[pd[20], a[2]], SEQ[f],
    SEQ[i[1], rr[0]]]],
  LRule[LEFT[], PRED[a[t_/;t<4]], RIGHT[], SUCC[a[1 + t]],
  ..
  LRule[LEFT[], PRED[rr[6]], RIGHT[], SUCC[rr[1]]],
  LRule[LEFT[], PRED[a[2]], RIGHT[], SUCC[STACK[i[4], YR[34]]],
  LRule[LEFT[], PRED[a[0]], RIGHT[], SUCC[STACK[YR[18], YR[18]]]]]
```

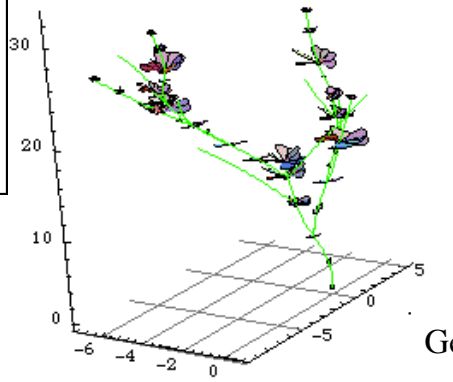
Duplication



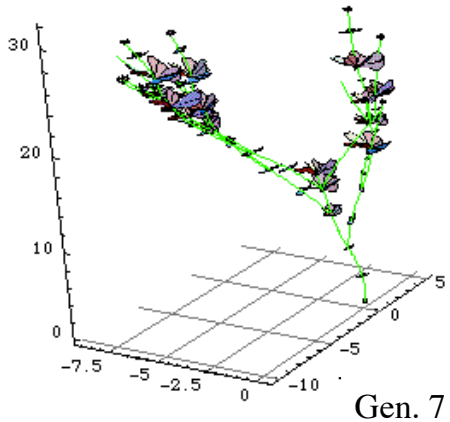
Gen. 8

```
LSystem[AXIOM[a[4]],
 LRULES{
  LRule[LEFT[], PRED[a[4]], RIGHT[],
   SUCC[SEQ[SEQ[f], SEQ[i[2],
    STACK[pd[60], l[0]], SEQ[rr[90]],
    STACK[pd[20], a[3]], SEQ[rr[90]],
    SEQ[i[0], RR[17], RR[17], l[2],
    STACK[pd[60], l[0]], SEQ[rr[90]],
    STACK[pd[20], a[2]], SEQ[f],
    SEQ[i[1], rr[0]]]],
  LRule[LEFT[], PRED[a[t_/;t<4]], RIGHT[], SUCC[a[1 + t]],
  ..
  LRule[LEFT[], PRED[rr[6]], RIGHT[], SUCC[rr[1]]],
  LRule[LEFT[], PRED[a[2]], RIGHT[], SUCC[STACK[i[4], i[4], YR[34]]],
  LRule[LEFT[], PRED[a[0]], RIGHT[], SUCC[STACK[YR[18], YR[18]]]]]
```

Gen. 4



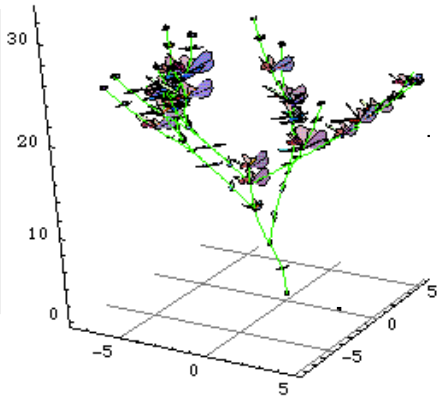
Gen. 6



Gen. 7

Effects of Genetic Operators

```
LSystem[AXIOM[a[4]],
 LRULES{
  LRule[LEFT[], PRED[a[4]], RIGHT[],
   SUCC[SEQ[SEQ[f], SEQ[i[2]],
    STACK[pd[60], l[0]], SEQ[rr[90]],
    STACK[pd[20], a[3]], SEQ[rr[90]],
    SEQ[i[0], RR[17], l[2]],
    STACK[pd[60], l[0]], SEQ[rr[90]],
    STACK[pd[20], a[2]], SEQ[f],
    SEQ[i[1], m[0]]],
  LRule[LEFT[], PRED[a[t_/;t<4]], RIGHT[], SUCC[a[1 + t]],
  ..
  LRule[LEFT[], PRED[m[6]], RIGHT[], SUCC[m[1]],
  LRule[LEFT[], PRED[a[2]], RIGHT[], SUCC[STACK[i[4], i[4], YR[34]]],
  LRule[LEFT[], PRED[a[0]], RIGHT[], SUCC[STACK[YR[18], YR[18]]]]]
```



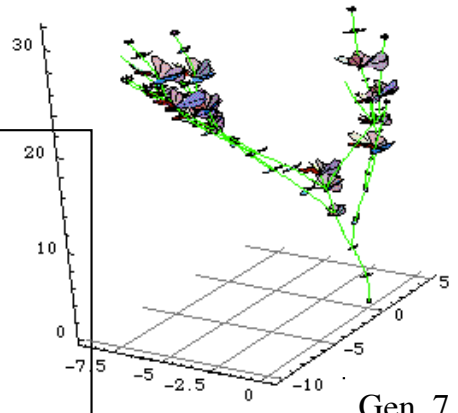
Gen. 8

Duplication

```
LSystem[AXIOM[a[4]],
 LRULES{
  LRule[LEFT[], PRED[a[4]], RIGHT[],
   SUCC[SEQ[SEQ[f], SEQ[i[2]],
    STACK[pd[60], l[0]], SEQ[rr[90]],
    STACK[pd[20], a[3]], SEQ[rr[90]],
    SEQ[i[0], RR[17], RR[17], l[2]],
    STACK[pd[60], l[0]], SEQ[rr[90]],
    STACK[pd[20], a[2]], SEQ[f],
    SEQ[i[1], m[0]]],
  LRule[LEFT[], PRED[a[t_/;t<4]], RIGHT[], SUCC[a[1 + t]],
  ..
  LRule[LEFT[], PRED[m[6]], RIGHT[], SUCC[m[1]],
  LRule[LEFT[], PRED[a[2]], RIGHT[], SUCC[STACK[i[4], i[4], YR[34]]],
  LRule[LEFT[], PRED[a[0]], RIGHT[], SUCC[STACK[YR[18], YR[18]]]]]
```

Duplication
Mutation

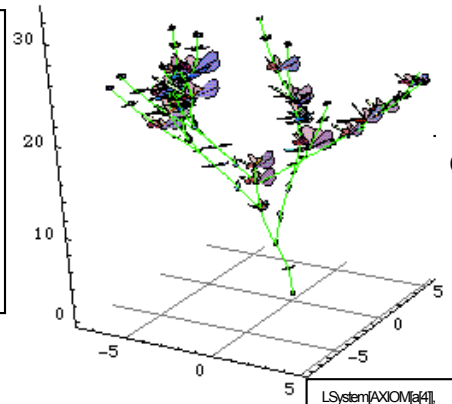
```
LSystem[AXIOM[a[4]],
 LRULES{
  LRule[LEFT[], PRED[a[4]], RIGHT[],
   SUCC[SEQ[SEQ[f], SEQ[i[2]],
    STACK[pd[60], l[0]],
    SEQ[SEQ[f], SEQ[i[2]],
    STACK[pd[60], l[0]], SEQ[rr[90]],
    STACK[pd[20], a[3]], SEQ[rr[90]],
    SEQ[i[3], m[4], RL[30]],
    STACK[pd[60], l[0]], SEQ[rr[90]],
    STACK[pd[20], a[2]], SEQ[f],
    SEQ[i[1], i[1], m[0]],
    STACK[pd[20], a[3]], SEQ[rr[90]],
    SEQ[i[0], RR[17], RR[17], l[2]],
    STACK[pd[60], l[0]], SEQ[rr[90]],
    STACK[pd[20], a[2]], SEQ[f],
    SEQ[i[1], m[0]]],
  LRule[LEFT[], PRED[a[t_/;t<4]], RIGHT[], SUCC[a[1 + t]],
  ..
  LRule[LEFT[], PRED[m[6]], RIGHT[], SUCC[m[1]],
  LRule[LEFT[], PRED[a[2]], RIGHT[], SUCC[STACK[i[4], i[4], YR[34]]],
  LRule[LEFT[], PRED[a[0]], RIGHT[], SUCC[STACK[YR[18], YR[18]]]]]
```



Gen. 7

Effects of Genetic Operators

```
LSystem[AXIOM[a[4]],
 LRULES{
  LRule[LEFT[], PRED[a[4]], RIGHT[],
   SUCC[SEQ[SEQ[f], SEQ[i[2]],
    STACK[pd[60], I[0]], SEQ[rr[90]],
    STACK[pd[20], a[3]], SEQ[rr[90]],
    SEQ[i[0], RR[17], I[2]],
    STACK[pd[60], I[0]], SEQ[rr[90]],
    STACK[pd[20], a[2]], SEQ[f],
    SEQ[i[1], m[0]]],
  LRule[LEFT[], PRED[a[t_/;t<4]], RIGHT[], SUCC[a[1 + t]],
  ..
  LRule[LEFT[], PRED[m[6]], RIGHT[], SUCC[m[1]],
  LRule[LEFT[], PRED[a[2]], RIGHT[], SUCC[STACK[i[4], i[4], YR[34]]],
  LRule[LEFT[], PRED[a[0]], RIGHT[], SUCC[STACK[YR[18], YR[18]]]]]
```



Gen. 8

Duplication

```
LSystem[AXIOM[a[4]],
 LRULES{
  LRule[LEFT[], PRED[a[4]], RIGHT[],
   SUCC[SEQ[SEQ[f], SEQ[i[2]],
    STACK[pd[60], I[0]], SEQ[rr[90]],
    STACK[pd[20], a[3]], SEQ[rr[90]],
    SEQ[i[0], RR[17], RR[17], I[2]],
    STACK[pd[60], I[0]], SEQ[rr[90]],
    STACK[pd[20], a[2]], SEQ[f],
    SEQ[i[1], m[0]]],
  LRule[LEFT[], PRED[a[t_/;t<4]], RIGHT[], SUCC[a[1 + t]],
  ..
  LRule[LEFT[], PRED[m[6]], RIGHT[], SUCC[m[1]],
  LRule[LEFT[], PRED[a[2]], RIGHT[], SUCC[STACK[i[4], i[4], YR[34]]],
  LRule[LEFT[], PRED[a[0]], RIGHT[], SUCC[STACK[YR[18], YR[18]]]]]
```

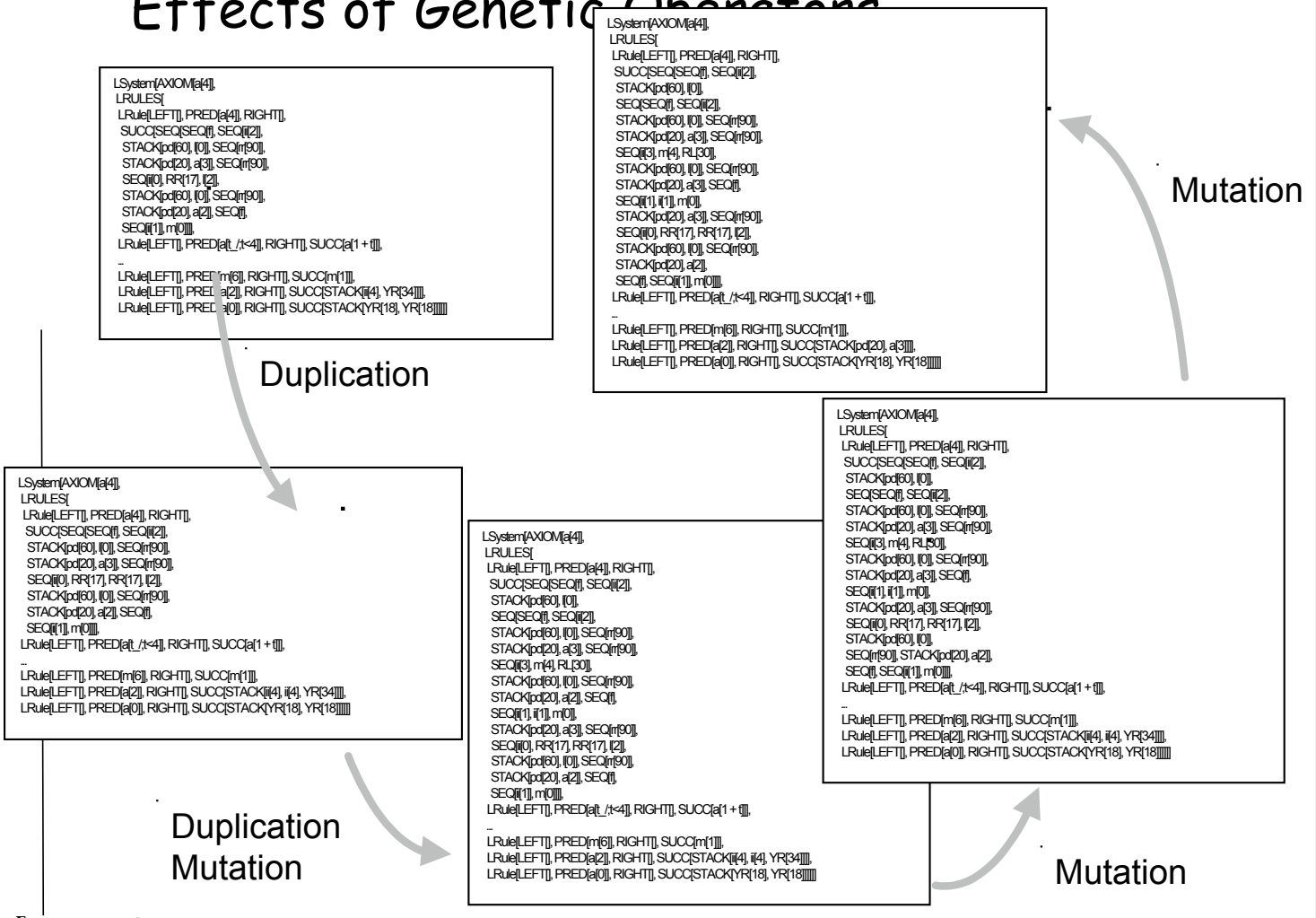
Duplication
Mutation

```
LSystem[AXIOM[a[4]],
 LRULES{
  LRule[LEFT[], PRED[a[4]], RIGHT[],
   SUCC[SEQ[SEQ[f], SEQ[i[2]],
    STACK[pd[60], I[0]],
    SEQ[SEQ[f], SEQ[i[2]],
    STACK[pd[60], I[0]], SEQ[rr[90]],
    STACK[pd[20], a[3]], SEQ[rr[90]],
    SEQ[i[3], m[4], RL[30]],
    STACK[pd[60], I[0]], SEQ[rr[90]],
    STACK[pd[20], a[2]], SEQ[f],
    SEQ[i[1], i[1], m[0]],
    STACK[pd[20], a[3]], SEQ[rr[90]],
    SEQ[i[0], RR[17], RR[17], I[2]],
    STACK[pd[60], I[0]], SEQ[rr[90]],
    STACK[pd[20], a[2]], SEQ[f],
    SEQ[i[1], m[0]]],
  LRule[LEFT[], PRED[a[t_/;t<4]], RIGHT[], SUCC[a[1 + t]],
  ..
  LRule[LEFT[], PRED[m[6]], RIGHT[], SUCC[m[1]],
  LRule[LEFT[], PRED[a[2]], RIGHT[], SUCC[STACK[i[4], i[4], YR[34]]],
  LRule[LEFT[], PRED[a[0]], RIGHT[], SUCC[STACK[YR[18], YR[18]]]]]
```

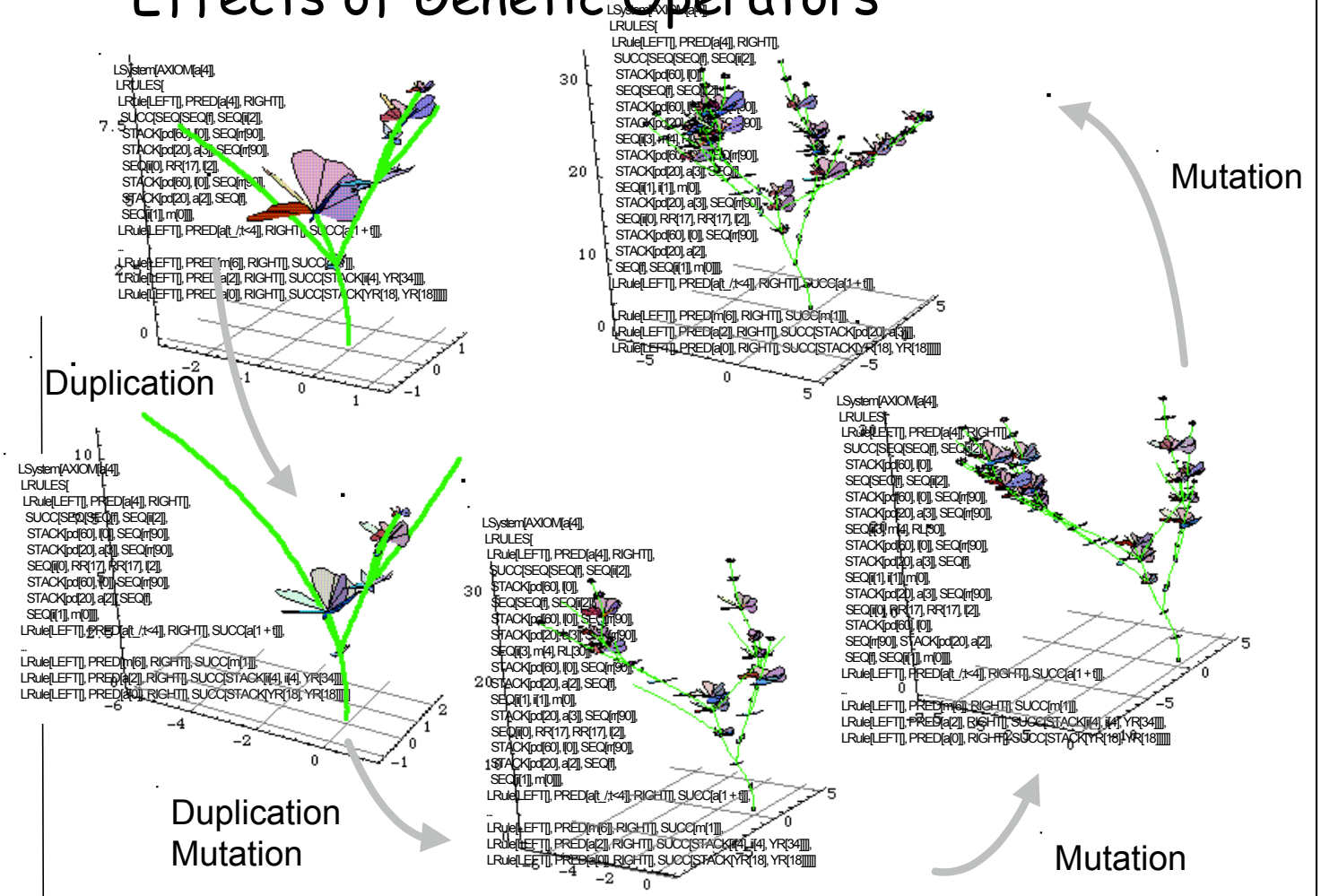
Mutation

```
LSystem[AXIOM[a[4]],
 LRULES{
  LRule[LEFT[], PRED[a[4]], RIGHT[],
   SUCC[SEQ[SEQ[f], SEQ[i[2]],
    STACK[pd[60], I[0]],
    SEQ[SEQ[f], SEQ[i[2]],
    STACK[pd[60], I[0]], SEQ[rr[90]],
    STACK[pd[20], a[3]], SEQ[rr[90]],
    SEQ[i[3], m[4], RL[50]],
    STACK[pd[60], I[0]], SEQ[rr[90]],
    STACK[pd[20], a[3]], SEQ[f],
    SEQ[i[1], i[1], m[0]],
    STACK[pd[20], a[3]], SEQ[rr[90]],
    SEQ[i[0], RR[17], RR[17], I[2]],
    STACK[pd[60], I[0]],
    SEQ[rr[90]], STACK[pd[20], a[2]],
    SEQ[f], SEQ[i[1], m[0]]],
  LRule[LEFT[], PRED[a[t_/;t<4]], RIGHT[], SUCC[a[1 + t]],
  ..
  LRule[LEFT[], PRED[m[6]], RIGHT[], SUCC[m[1]],
  LRule[LEFT[], PRED[a[2]], RIGHT[], SUCC[STACK[i[4], i[4], YR[34]]],
  LRule[LEFT[], PRED[a[0]], RIGHT[], SUCC[STACK[YR[18], YR[18]]]]]
```

Effects of Genetic Operators



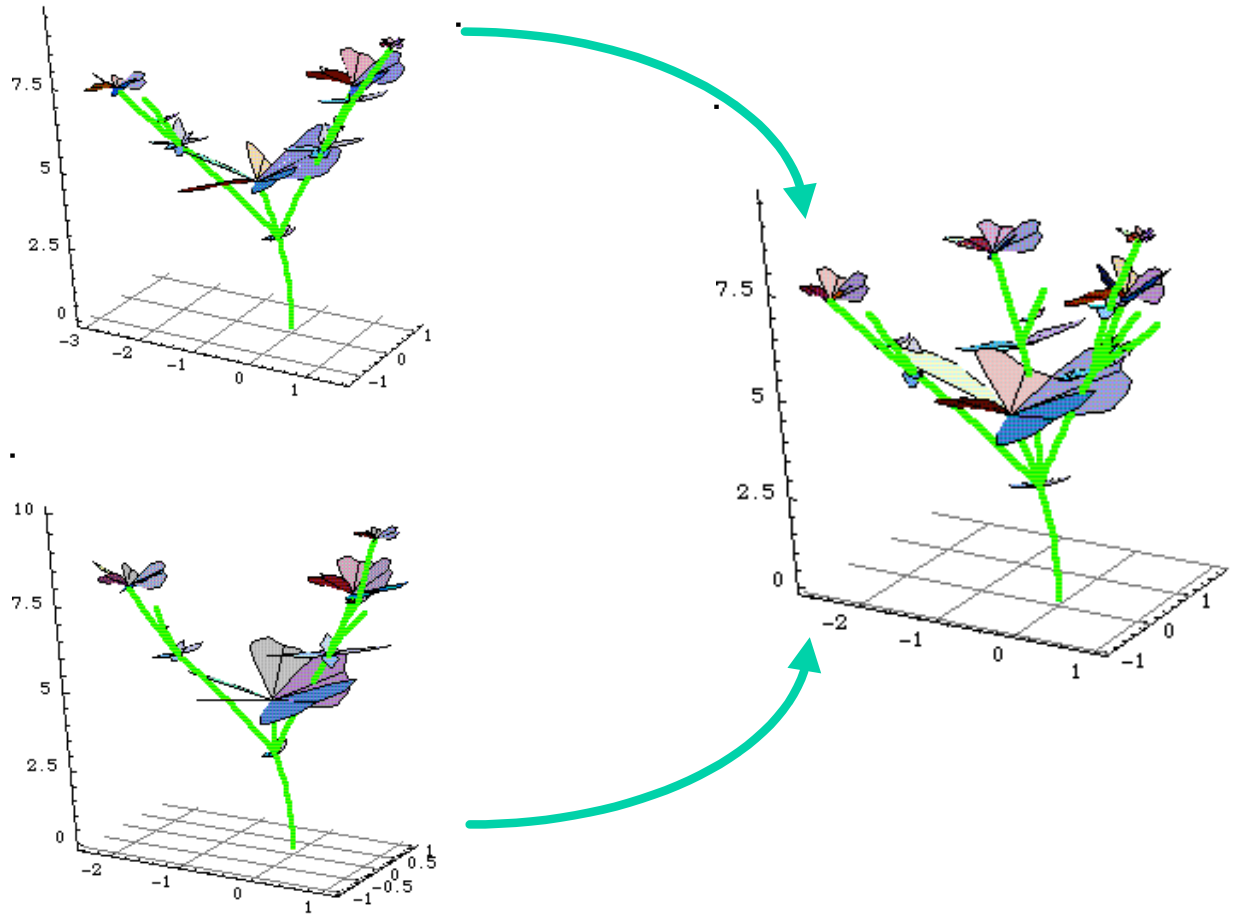
Effects of Genetic Operators



EVOLUTIONARY Swarm Design

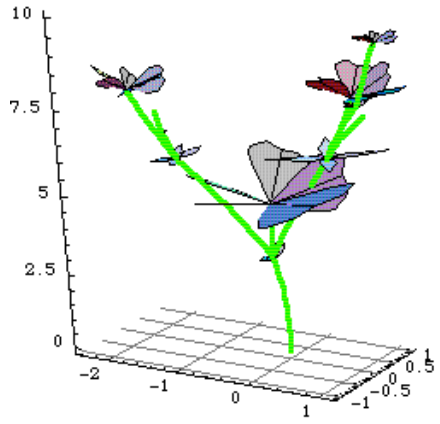
Christian Jacob, University of Calgary

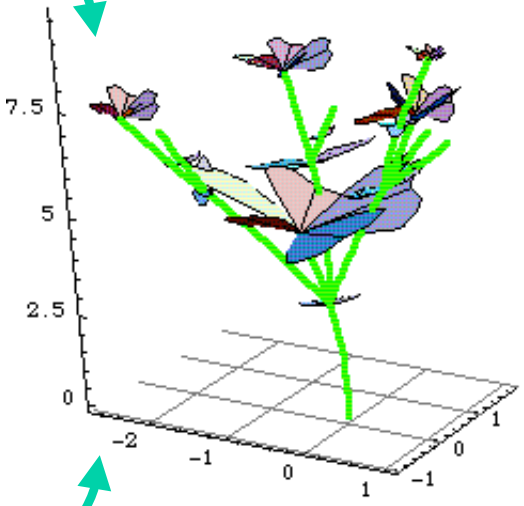
Effects of Recombination



Effects of Recombination

```
LSystem[AXIOM[a[4]],
 LRULES[
  LRule[LEFT[], PRED[a[4]], RIGHT[],
   SUCC[SEQ[SEQ[f], SEQ[ii[2]],
   STACK[pd[20], a[2]] , SEQ[rr[90]],
    STACK[pd[20], a[3]], SEQ[rr[90]],
    SEQ[l[2]], STACK[pd[60], l[0]],
    SEQ[rr[90]], STACK[pd[20], a[2]],
    SEQ[f], SEQ[ii[1]], m[0]]]],
...]
```

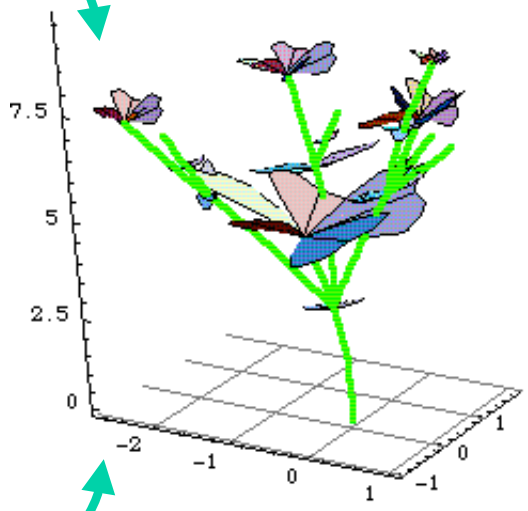




Effects of Recombination

```
LSystem[AXIOM[a[4]],
 LRULES[
  LRule[LEFT[], PRED[a[4]], RIGHT[],
   SUCC[SEQ[SEQ[f], SEQ[ii[2]],
   STACK[pd[20], a[2]] , SEQ[rr[90]],
    STACK[pd[20], a[3]], SEQ[rr[90]],
    SEQ[l[2]], STACK[pd[60], l[0]],
    SEQ[rr[90]], STACK[pd[20], a[2]],
    SEQ[f], SEQ[ii[1]], m[0]]]],
 ...]
```

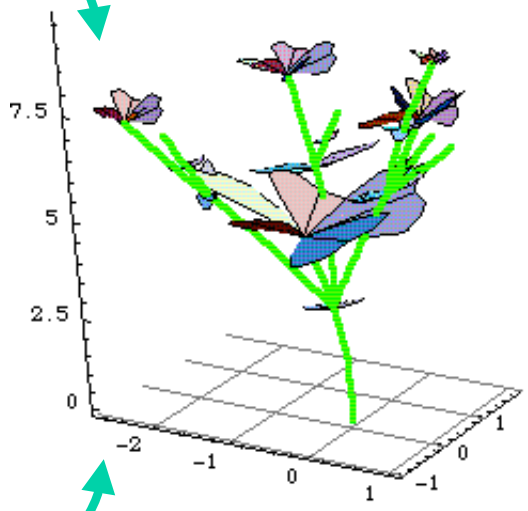
```
LSystem[AXIOM[a[4]],
 LRULES[
  LRule[LEFT[], PRED[a[4]], RIGHT[],
   SUCC[SEQ[SEQ[f], SEQ[ii[2]],
   STACK[pd[60], l[0]], SEQ[rr[90]],
   STACK[pd[20], a[3]], SEQ[rr[90]],
   SEQ[YR[10]], STACK[pd[60], l[0]],
   SEQ[rr[90]], STACK[pd[60], l[0]],
   SEQ[f], SEQ[ii[1]], m[0]]]],
 ...]
```



Effects of Recombination

```
LSystem[AXIOM[a[4]],
 LRULES[
  LRule[LEFT[], PRED[a[4]], RIGHT[],
   SUCC[SEQ[SEQ[f], SEQ[ii[2]],
   STACK[pd[20], a[2]] , SEQ[rr[90]],
    STACK[pd[20], a[3]], SEQ[rr[90]],
    SEQ[l[2]], STACK[pd[60], l[0]],
    SEQ[rr[90]], STACK[pd[20], a[2]],
    SEQ[f], SEQ[ii[1]], m[0]]]],
 ...]

LSystem[AXIOM[a[4]],
 LRULES[
  LRule[LEFT[], PRED[a[4]], RIGHT[],
   SUCC[SEQ[SEQ[f], SEQ[ii[2]],
   STACK[pd[60], l[0]], SEQ[rr[90]],
   STACK[pd[20], a[3]], SEQ[rr[90]],
   SEQ[YR[10]], STACK[pd[60], l[0]],
   SEQ[rr[90]], STACK[pd[60], l[0]],
   SEQ[f], SEQ[ii[1]], m[0]]]],
 ...]
```



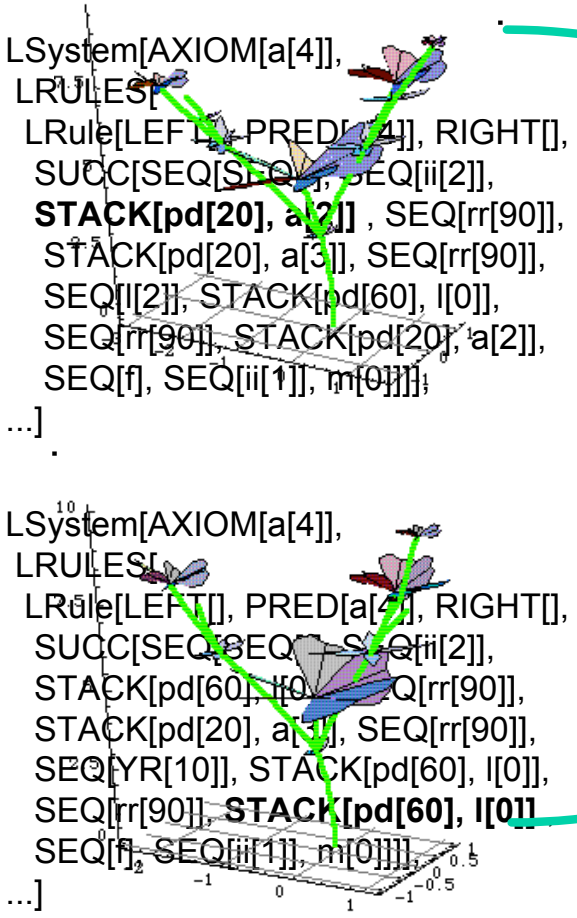
Effects of Recombination

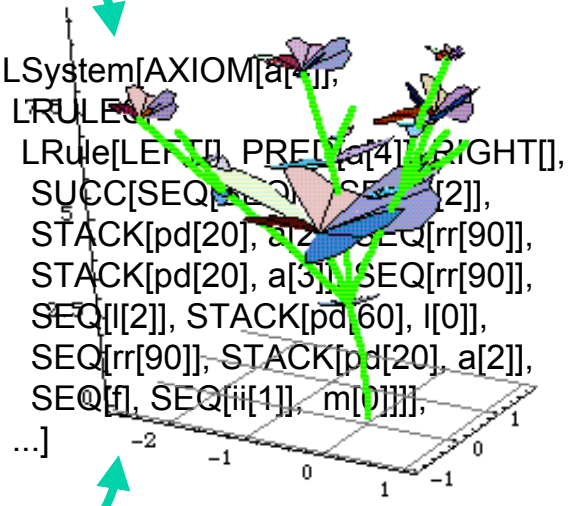
```
LSystem[AXIOM[a[4]],
 LRULES[
  LRule[LEFT[], PRED[a[4]], RIGHT[],
   SUCC[SEQ[SEQ[f], SEQ[ii[2]],
   STACK[pd[20], a[2]] , SEQ[rr[90]],
    STACK[pd[20], a[3]], SEQ[rr[90]],
    SEQ[l[2]], STACK[pd[60], l[0]],
    SEQ[rr[90]], STACK[pd[20], a[2]],
    SEQ[f], SEQ[ii[1]], m[0]]]],
...]
```

```
LSystem[AXIOM[a[4]],
 LRULES[
  LRule[LEFT[], PRED[a[4]], RIGHT[],
   SUCC[SEQ[SEQ[f], SEQ[ii[2]],
   STACK[pd[60], l[0]], SEQ[rr[90]],
   STACK[pd[20], a[3]], SEQ[rr[90]],
   SEQ[YR[10]], STACK[pd[60], l[0]],
   SEQ[rr[90]], STACK[pd[60], l[0]],
   SEQ[f], SEQ[ii[1]], m[0]]]],
...]
```

```
LSystem[AXIOM[a[4]],
 LRULES[
  LRule[LEFT[], PRED[a[4]], RIGHT[],
   SUCC[SEQ[SEQ[f], SEQ[ii[2]],
   STACK[pd[20], a[2]], SEQ[rr[90]],
   STACK[pd[20], a[3]], SEQ[rr[90]],
   SEQ[l[2]], STACK[pd[60], l[0]],
   SEQ[rr[90]], STACK[pd[20], a[2]],
   SEQ[f], SEQ[ii[1]],  m[0]]]],
...]
```

Effects of Recombination





References

Genetic Programming:

- Jacob, C. (2001). Illustrating Evolutionary Computation with Mathematica. San Francisco, CA, Morgan Kaufmann Publishers (Chapters 7 and 8).
- Koza, J. R. (1992). Genetic Programming: On the Programming of Computers by Means of Natural Selection. Cambridge, MA, MIT Press.
- Koza, J. R. (1994). Genetic Programming II: Automatic Discovery of Reusable Programs. Cambridge, MA, MIT Press.
- Koza, J. R., D. Andre, et al. (1998). Genetic Programming III: Automatic Programming and Automatic Circuit Synthesis. San Francisco, CA, Morgan Kaufmann.

Lindenmayer Systems:

- Jacob, C. (2001). Illustrating Evolutionary Computation with Mathematica. San Francisco, CA, Morgan Kaufmann Publishers (Chapters 7 and 8).
- Prusinkiewicz, P. and A. Lindenmayer (1990). The Algorithmic Beauty of Plants. New York, Springer.