Demonstration of the GroIMP software

Ole Kniemeyer, Gerhard Buck-Sorlin, Winfried Kurth

{okn,wk}@informatik.tu-cottbus.de

buck@ipk-gatersleben.de.

Brandenburgische Technische Universität Cottbus

Department of Computer Science

Chair for Practical Computer Science / Graphics Systems

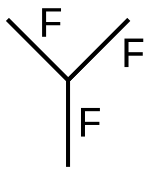
Funded by Deutsche Forschungsgemeinschaft, Research Unit *Virtual Crops*

▸ Overview

Features of the GroIMP software:

- L-system modelling
- User interaction
- Networking
- Java implementation
- Graph grammar modelling
- Network modelling
- Open data model

▸ L-system modelling

- Data structure: Linear *string* of symbols, e.g.,
  F [+ F] [- F]
- Turtle graphics interpretation leads to geometrical structures: 

- String replacement rules implement dynamics:
  A $\longrightarrow$ F [+ A] [- A]
  They are applied in parallel.
- Plant growth can be suitably described by such a *rule-based* process.
- Realistic images can be produced.

▸ A 2D tree stand model

- Monopodial growth: `X` $\longrightarrow$ `F [+X] [-X] X`
- Phototropism: Shoots bend towards a light source.
- Growth condition: A light cone emerging from the tip of a meristem has to be free.
- Reproduction: Production of seeds, spreading, germination.

Implementation in *XL* is straightforward.

▶ Tree stand model: Implementation

■ Monopodial growth ■ Parametrization ■ Growth termination ■ Phototropism ■ Shading

```
x:X(r, l) ==>
  if(r == 2) (
    if (!isShaded(x)) (
      tropism(x, sun.basis, 0.2f)
      F(l, 0.02f) Leaf
    )
  ) else if ((l > minLength[r]) && !isShaded(x)) (
    tropism(x, sun.basis, 0.2f) F(l, 0.02f)
    [RU(angle[r]) X(r+1, l*c2)]
    [RU(-angle[r]) X(r+1, l*c2)]
    X(r, l*c1)
  );
```

▶ Tree stand model: Reproduction

■ Seed production ■ Seed spreading ■ Germination

```
n:Leaf, (random(0, 1) < 0.005) ==>>
  n, ˆ Seed(getGlobalOrigin(n));

Seed(b) ==>>
  if (b.z <= 0) (
    {b.z = 0;}
    ˆ TranslationNode(b) Tree(0) X(0, 1)
  ) else {
    b.x += random(-1, 1);
    b.z -= random(0.1, 0.3);
    break;
  };
```

Software demonstration I

- Simulation of tree stand model within GroIMP
- Object inspection
- User interaction: Tree cutting
- User interaction: Movement of light source
- Networking

Résumé I

- Models written in XL can be simulated within GroIMP.
- “Symbols” are real Java objects.
- Methods can be defined in XL:

```
boolean isShaded(Node s) {...}
```

- Existing Java methods can be used:

```
intersectsFrustum(f, s, 40*DEG, 0.05, 1.1)
```

- Global queries can be formulated easily:

```
exist((* f:F, ((f != s) && ...)  *))
```

▸ Implementation of GroIMP/XL I

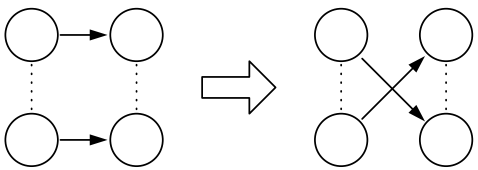
GroIMP and XL are implemented in Java.

- All available Java runtime libraries are accessible within XL.
- By standard Java mechanisms, GroIMP/XL can be coupled with non-Java software.
- Some software systems (e.g., MATLAB) have a direct Java integration – hence a direct GroIMP/XL integration.
- There exist native Java compilers (e.g., `gcj`) which can combine Java and C code.

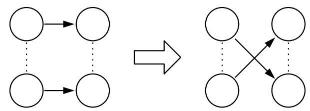
▸ From strings to graphs

Numerous processes in biology can be described more concisely using graphs instead of strings.

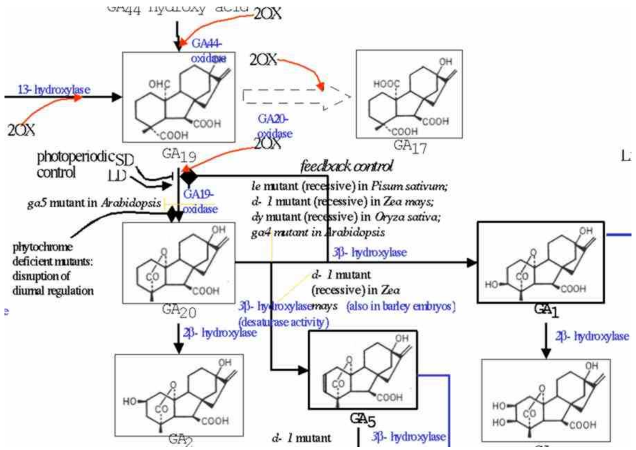
- Crossing over of two genomes:



⇒ L-system rule?

⇒ Graph rewriting rule 

- Metabolic or gene regulatory network simulation:



⇒ L-system string encoding?

⇒ Representation as graph

Relational Growth Grammars

Relational Growth Grammars (RGG) extend the established concept of L-systems:

- Graphs instead of strings
- Graph rewriting instead of string rewriting
- Objects instead of symbols
- Edges and relations instead of string neighbourhood
- Multiple scales representable by specific edges
- Free mixing of rule-based and imperative programming

▸ XL: An implementation of RGG

XL is a Java-based implementation of RGG for the use in practice.

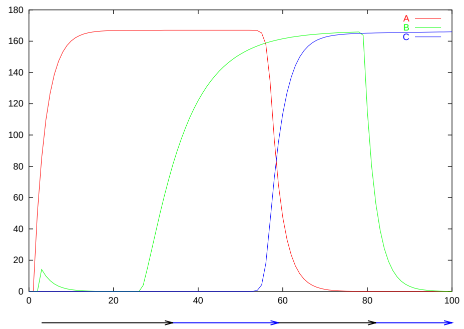
- Imperative Java constructs (classes, methods, variables, loops, ...)
- Graph rewriting rules and queries
- Integration in GroIMP
- Certain GroIMP Java classes as turtle commands

RGG/XL extend capabilities of L-systems towards *functional*-structural modelling in an integrative way.

▸ ABC model

The ABC model predicts flower morphogenesis on the basis of a genetic regulatory network.

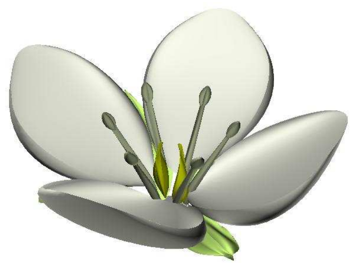
- Three genes, A, B and C
- Transcription factors determine type of flower organ to be formed.
- Factor concentrations change in time.



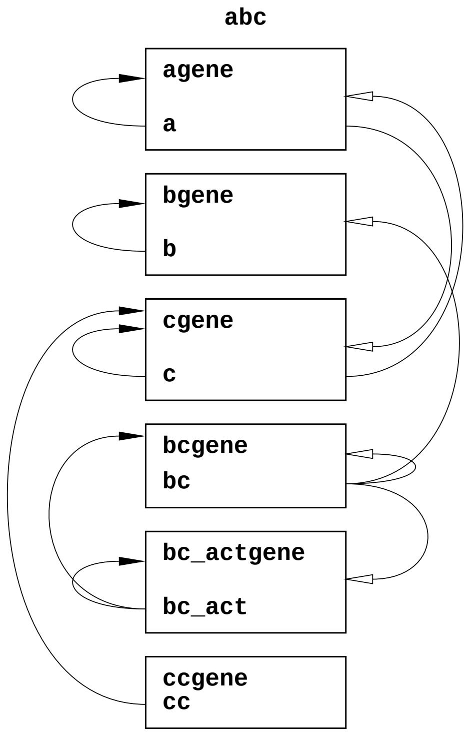
$\Rightarrow$

$$
\begin{array}{rcl}
[a] > 80; [b], [c] < 80 & \longrightarrow & \text{sepal} \\
[b] > 80, [c] \leq [a] & \longrightarrow & \text{petal} \\
 & \ldots &
\end{array}
$$

$\Rightarrow$



▸ Model and implementation details



Network [a]

→ Activation of gene
▷ Repression of gene

Quantification: Michaelis-Menten equation $V = \frac{V_{\max} c_f}{c_f + K_m}$

- Construction of network

```
agene:Gene(0.1) -encodes-> a:Factor(0, 0.3),
...,
c Activate(50, -100) agene, ...
```

- Michaelis-Menten kinetics

```
p:Factor <-encodes- g:Gene(v0) ::>
  p.concentration :+= max(0,
    sum(((* Factor(cf,) Activate(km, max) g *),
             max*cf / (cf+km))) + v0);
```

[a] This model is an XL translation of a model by Jan T. Kim.

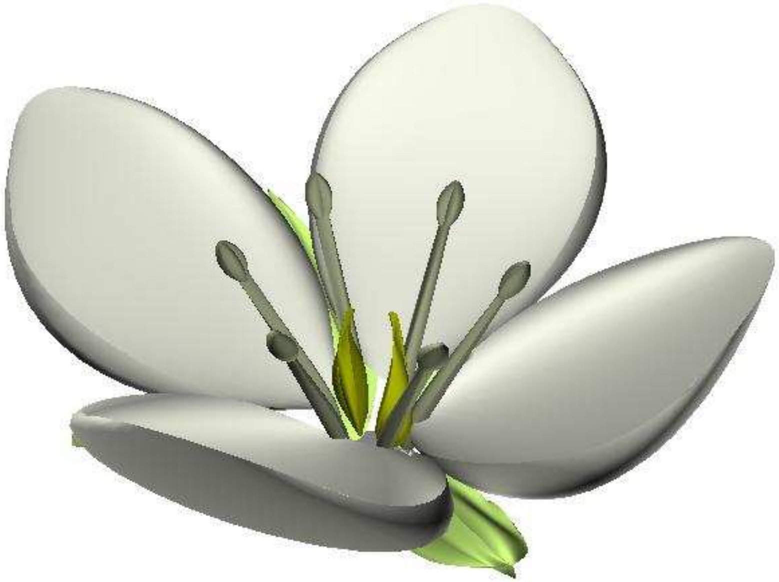
▸ Implementation of flower morphology

- Flower morphogenesis follows simple scheme
  `Meristem → Organ [Lateral]... Meristem.`
- Type and parameters of flower organs to be formed controlled by transcription factor concentrations

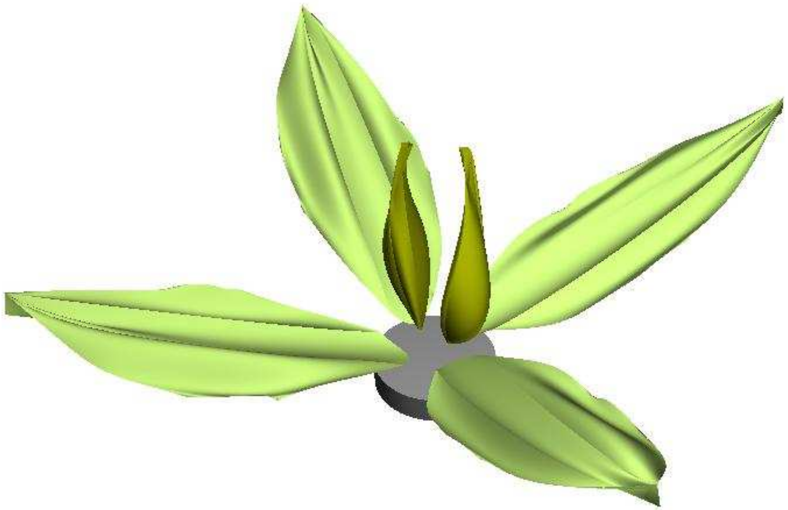
```
m:Meristem(type, mass)
  (* -factors-> Factor(a,) Factor(b,) Factor(c,) Factor(cc,) *) ==>
  {
    int t = ((c > 80) && (cc > 1)) ? TERMINATE
            : (b > 80) ? ((c > a) ? STAMEN : PETAL) ...;
  }
  if (t == type) {m.mass++; break;} else (
    if (type == SHOOT)       (F(0.45*mass))
    else if (type == SEPAL) (F(0.1) [sepal(0)] [sepal(2)]
                                    [sepal(4)] [sepal(6)])
    else if (type == ...)   (...)
    if (t != TERMINATE)     (m(t, 1))
  );
```

▶ Software demonstration II

- Simulation of ABC model
- “Mutation” of source code: Modifying the network
- Simulation of mutants



Wild type



“Loss of B” mutant

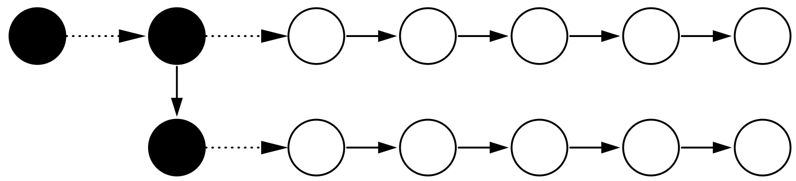
▶ Hordeomorphs

Model of genotype-phenotype relationship using RGG

- Virtual creatures resembling barley ears
- Idea based on R. Dawkins' "biomorphs"
- Diploid genome of five genes
- Genetic operations mutation, selection by user, asexual reproduction, sexual reproduction
- Morphology is modelled in an L-system style, controlled by genome

Hordeomorph implementation

- Genome representation

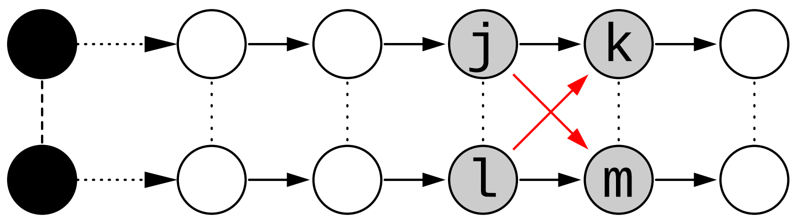


```
Genome -first-> Chromo [-first-> 1 1 0 1 0]
                Chromo [-first-> 1 0 0 0 0]
```

- Mutation

```
int ==> if (prob(0.3)) irandom(0, 1) else break;
```

- Crossing over




```
int j, k, l, m;
j k, l m, j -align- l, (* higher(j) -mate- higher(l) *)
    ==>> j m, l k
```

▸ Software demonstration III

- Simulation of Hordeomorph model
- User selection: Asexual reproduction
- User selection: Sexual reproduction

▸ A barley model

- Morphogenesis is modelled in an L-System style.
- Diploid genome controls ear morphogenesis.
- Metabolic network (part of Gibberellic acid biosynthesis) in each internode organ controls internode elongation:

```
Cell [s:GA20] [p:GA1] ::> michaelisMenten(s, p, 0.2, 1);
...
i:Internode [s:GA1] ::> i.length :+= DT * C * s.concentration;

void michaelisMenten(Substance s, Substance p, double max, double km) {
    double r = DT * max * s.concentration / (km + s.concentration);
    s.concentration :+= -r;
    p.concentration :+= r;
}
```

- Transport of metabolites

▸ Software demonstration IV

- Simulation of barley model

Wild type

Dwarf

Slender

▶ Implementation of GroIMP/XL II

At runtime, graphs are inspected and modified through a graph data model interface.

- Default data model establishes link between GroIMP objects and XL runtime library.
- Other data model implementations may enable XL to operate on other data structures.
- Data model implementation for commercial 3D-modelling software CINEMA 4D (MAXON) is in the works.

Résumé II

RGGs provide a concise way of implementing biological models:

- Fundamental data structure is a graph.
- Complex relationships can be represented as a graph.
- Needs of functional modelling are addressed.
- The structural view of L-systems is preserved.

Outlook I

- Runtime efficiency is crucial
  - Graph grammars introduce runtime overhead
  - Improvement of matching algorithm
  - Java byte-code generation
- Enhancement of 3D-visualization
  - Java 3D
  - External 3D-rendering tools
  - Smooth animation
- Improvement of workflow

▶ Outlook II

Data interfaces

- Reintegration of VRML, POV-Ray, MTG
- Digital Elevation Model (DEM)

Binary interfaces

- CINEMA 4D (MAXON)
- Delphi (Borland)

Publishing of GroIMP

- Website www.grogra.de
- Software will be made open-source