Modeling Growth
with
L-Systems & *Mathematica*

The symbolic and graphic capabilities of Mathematica are used to implement and visualize parallel rewrite systems.

by Christian Jacob

Rewriting has proved to be a useful technique for defining complex objects by successively replacing parts of simple initial objects using a set of rewrite rules or productions. Rewriting systems operating on character strings have been successfully used for describing syntactic features of natural languages or for formal definitions of programming languages [Cho56], [Bac59].

Here we want to focus on a special type of rewrite systems, commonly termed *L-systems*, which are used in theoretical biology in order to describe and simulate natural growth processes. The introduction of L-systems dates back until 1968 when the biologist Aristid Lindenmayer (hence L-systems) defined a formal rule system (production system) where all letters in a given word are replaced in parallel and simultaneously [Lin68]. This feature makes L-systems especially suitable for desribing fractal structures, cell divisions in multicellular organisms or flowering stages of herbaceous plants [Pru90].

Formal definition of L-systems

Context-free L-systems

D0L-systems (D0 means deterministic with no context) are the simplest type of L-systems. Formally a D0L-system can be defined as a triple G = (Σ,P,α) where $\Sigma=\{s_1, s_2, ..., s_n\}$ is an alphabet, α, referred to as the axiom, is an element of Σ^*, the set of all finite words over alphabet Σ.

The structure preserving mapping P is defined by a production map $P:\Sigma\text{->}\Sigma^*$ with s -> P(s) for each s in Σ. As we consider only deterministic L-systems there is exactly one production rule for each symbol s. The word sequence E(G) = α(0), α(1) , α(2), ... generated by G is derived as follows:

$$\alpha(0) = P[0](\alpha) = \alpha,\ \alpha(1) = P[1](\alpha),\ \alpha(2) = P[2](\alpha), \ldots$$

where P[i] (α) denotes i-fold application of P and where each symbol α(i+1) is obtained from the preceding string

$$\alpha(i) = \alpha_1(i)\ \alpha_2(i) \ldots \alpha_m(i)$$

by applying the production rules to all *m* symbols of the string simultaneously:

$$\alpha(i+1) = P(\alpha_1(i))\ P(\alpha_2(i)) \ldots P(\alpha_m(i))$$

The language L(G) of G is defined by $L(G) = \{ P[i](\alpha) \mid i \geq 0 \}$.
We will demonstrate the usefulness of this formal definition for the description of growth processes with some little examples. Let us have a closer look at how to simulate development of multicellular filaments. The following growth process can be observed with various algae and especially in the blue-green bacteria Anabaena catenula [Pru89].
Suppose that we want to represent two cytological states, termed *a* and *b*, of the cells which characterize their size and readiness to divide. The subscripts *l* and *r* are indicative of cell polarity, specifying the positions - left or right - in which daughter cells of type *a* and *b* will be produced.

The following L-system describes the development of a filament:

Axiom: a_r
p_1: a_r -> $a_l b_r$
p_2: a_l -> $b_l a_r$
p_3: b_r -> a_r
p_4: b_l -> a_l

An *a*-cell with right orientation (a_r) divides into another *a*-cell with opposite orientation and a *b*-cell oriented to the right (production p_1). An analogous interpretation holds for production p_2. With p_3 and p_4 a *b*-cell converts into an according *a*-cell.
Starting with the axiom string this rewrite system generates the following sequence of words:

a_r
$a_l b_r$
$b_l a_r a_r$
$a_l a_l b_r a_l b_r$
$b_l a_r b_l a_r a_r b_l a_r a_r$
...

In each step every symbol a_r, a_l, b_r or b_l matching a predecessor (the left side of the rule) is replaced by the according successor (the right side string of the rule). This replacement is done simultaneously within each generation step.
Before we show how to implement these concepts in *Mathe- matica we will have a brief look at the (biologically more relevant) extension to L-systems rules depending on context.*

Context-sensitive L-systems

Up to now we have discussed context-independent rewriting, i.e. the way a letter is rewritten depends on the letter only, adjacent letters have no influence on the rewrite process. However, if there is a need to simulate interaction (e.g. of cells within a layer) context-sensitive L-systems, commonly known as *IL-systems*, have to be used.

In the most general definition of IL-systems the rewriting of a letter depends on *m* of its left and *n* of its right neighbors, where *m* and *n* are fixed integers. These systems are denoted as (*m*,*n*)L-systems which resemble context-sensitive Chomsky- grammars, but - as L-system rewriting is parallel in nature - every symbol is rewritten in each derivation step; this is especially important whenever there is an overlap of context strings.

In order to make the following examples easier for demonstration we will focus on (1,1)L-systems (or 2L-systems) in the sequel. This means that each rule of IL-systems we will discuss has the form

l < p > r -> s

with l, p, r and s denoting *left context*, *predecessor*, *right context* and *successor*, respectively. The symbols "<" and ">" only separate context and predecessor strings. Thus an 0L-system as discussed in the previous section consists of rules with no left or right context of the form

< p > -> s

or simply

p -> s.

How does it look in *Mathematica*?

Now let us examine how we can represent L-systems in *Mathematica*. We first load the notebook package *kLSystems.ma*

```
<< kLSystems.ma
```

The *kLSystems* package contains defintions for the application of parallel rewrite rules of L-systems with left and right contexts with arbitrary length. Each rule of the form l < p > r -> s as described above is represented by a *Mathematica* expression of the form

LRule[LEFT[l], PRED[p], RIGHT[r], SUCC[s]].

Accordingly, we define the production set as an LRULES expression

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

and an L-system is described as follows:

LSystem[Axiom[...], LRULES[...]].

With this representation we can easily derive the type of the expressions and subexpressions by only looking at their head symbols.

This notation leads to the following description of the example L-sytem for *Anabaena catenula* presented in the previous section:

```
axiom = AXIOM[ aR ];

lrules =
  LRULES[
    LRule[LEFT[], PRED[ aR    ], RIGHT[],
                  SUCC[ aL,bR ]],
    LRule[LEFT[], PRED[ aL    ], RIGHT[],
                  SUCC[ bL,aR ]],
    LRule[LEFT[], PRED[ bR    ], RIGHT[],
                  SUCC[ aR    ]],
    LRule[LEFT[], PRED[ bL    ], RIGHT[],
                  SUCC[ aL    ]]
  ];

lsystem = LSystem[axiom,lrules];

skipPattern = {Null};
```

The **skipPattern** list includes all symbols that should not be considered as context. However, these skip patterns will not be used in the following examples.

In order to get formatted output of the resulting strings we define formatting patterns for the L-system expressions:

```
Format[AXIOM[x___]] := SequenceForm[x]

Format[aR] := Subscripted[a[r]]
Format[aL] := Subscripted[a[l]]
Format[bR] := Subscripted[b[r]]
Format[bL] := Subscripted[b[l]]

Format[LRule[LEFT[l___],PRED[p___],
             RIGHT[r___],SUCC[s___]]] :=
  SequenceForm[l," < ",p," > ",r," -> ",s]

Format[LSystem[a_,l_]] :=
  TableForm[{
    SequenceForm["Axiom: ",a],
    SequenceForm["Rules: ",
      ColumnForm[Apply[List,l]]]
  }]
```

This enables us to display our rewrite rules in the notation commonly used in L-system literature:

```
lsystem
```

```
Axiom: a
        r

Rules:  < a  >  -> a b
           r        l r
        < a  >  -> b a
           l        l r
        < b  >  -> a
           r        r
        < b  >  -> a
           l        l
```

Now we are ready to start an L-system simulation. The function *runKLSystem[lsys_LSystem,n_Integer]* takes an L-system expression *lsys* and the number *n* of rewrite steps as arguments and performs parallel expression rewriting starting with the axiom expression of *lsys*.

```
anabaenaGrowth = runKLSystem[lsystem, 6]
```

$\{a_r,\ a_l b_r,\ b_l a_r a_r,\ a_l a_l b_r a_l b_r,\ b_l a_r b_l a_r a_r b_l a_r a_r,$
$a_l a_l b_r a_l a_l b_r a_l b_r a_l a_l b_r a_l b_r,$
$b_l a_r b_l a_r a_r b_l a_r b_l a_r a_r b_l a_r a_r b_l a_r b_l a_r a_r b_l a_r a_r\}$

This output does not resemble growing cell layers, but the following formatted outputs are a nice, however very simple, visualization:

```
ColumnForm[%]
```

a_r
$a_l b_r$
$b_l a_r a_r$
$a_l a_l b_r a_l b_r$
$b_l a_r b_l a_r a_r b_l a_r a_r$
$a_l a_l b_r a_l a_l b_r a_l b_r a_l a_l b_r a_l b_r$
$b_l a_r b_l a_r a_r b_l a_r b_l a_r a_r b_l a_r a_r b_l a_r b_l a_r a_r b_l a_r a_r$

```
ColumnForm[%%,Center]
```

```
                     a
                      r
                    a b
                     l r
                   b a a
                    l r r
                 a a b a b
                  l l r l r
              b a b a a b a a
               l r l r r l r r
         a a b a a b a b a a b a b
          l l r l l r l r l l r l r
b a b a a b a b a a b a a b a b a a b a a
 l r l r r l r l r r l r r l r l r r l r r
```

Please note that *runKLSystem[lsys,n]* returns a list of expressions with head *AXIOM*, the *Format* definition for the *AXIOM* expressions strip off the head.

L-System string interpretation

The following definitions tell how the generated strings are interpreted as cell layers. As a side effect the interpretation function generates graphics objects that are used for visualization.

```
ClearAll[interpret];

interpret[AXIOM[z__]] :=
Module[{x = 0, y = 0},
  First[Take[Fold[interpret[#1,#2]&,
    {x,y,{}},{z}],-1]]
];

interpret[{x_,y_,l_}, aL ] :=
  {x+2,y,Append[l,ArrowedCircle[{x+1, y}, {1,.5},
                  Direction -> Left]]};
interpret[{x_,y_,l_}, aR ] :=
  {x+2,y,Append[l,ArrowedDisk[{x+1, y}, {1,.5},
                  Direction -> Right]]};
interpret[{x_,y_,l_}, bL ] :=
  {x+1,y,Append[l,ArrowedCircle[{x+.5, y}, {.5,.5},
                  Direction -> Left]]};
interpret[{x_,y_,l_}, bR ] :=
  {x+1,y,Append[l,ArrowedDisk[{x+.5, y}, {.5,.5},
                  Direction -> Right]]};
```

The interpretation for each of the symbols *aL*, *aR*, *bL* and *bR* is defined by the function *interpret[{x,y,l},symbol]* which first of all receives the x- and y-coordinates of the current cell and returns a coordinate pair for the following cell. The y-coordinate remains unchanged as we only consider layers growing horizontally but not vertically. Through the *l*-parameter a supplemented list of graphics objects (*ArrowedCircle*, *ArrowedDisk*) is passed on. This list is returned by the *AXIOM* interpretation function which takes only one argument, keeps the coordinates as local variables and folds *interpret* over each subexpression of the *AXIOM* term.

```
Needs["Graphics`Arrow`"];

ArrowedDisk[{x_,y_},{r1_,r2_},opts___] :=

 Graphics[{
   Thickness[0.001],
   GrayLevel[(1 - HueValue) /. {opts} /. Options[ArrowedDisk]],
   Disk[{x,y},{r1,r2}],Hue[0],

   If[(Direction /. {opts}) === Left,
     Arrow[{x+ .8 r1,y},{x- .8 r1,y},
       HeadScaling -> Relative],
     Arrow[{x- .8 r1,y},{x+ .8 r1,y},
       HeadScaling -> Relative]
   ],

   Hue[0.75],
   Circle[{x,y},{r1,r2}]
  }]
```

```
Options[ArrowedDisk] :=
{
  HueValue -> 0.1,
  Direction -> Right
}
```

```
ArrowedCircle[coords:{_,_},radii:{_,_},opts___] :=
  ArrowedDisk[coords,radii,opts,HueValue -> 0]
```

The listing above shows the definitions of the *ArrowedDisk* and *ArrowedCircle* graphics objects. After mapping the interpretation function on the L-system generated expressions the resulting graphics can be easily animated which gives instant insight into the simulated growth process. An auxiliary single point is put at the maximum horizontal coordinate in order to ensure all graphics keeping the same length.

```
disksAndCircles = Map[interpret,anabaenaGrowth];

maxX = Max[Cases[disksAndCircles,Circle[x___],
           Infinity] /. Circle[{x_,_},_] :> x];
Map[
  Show[Graphics[#],Graphics[{Hue[0],
    Point[{maxX+2,0}]}],
    AspectRatio -> Automatic,
    PlotRange -> All
  ]&,
  disksAndCircles
];
```

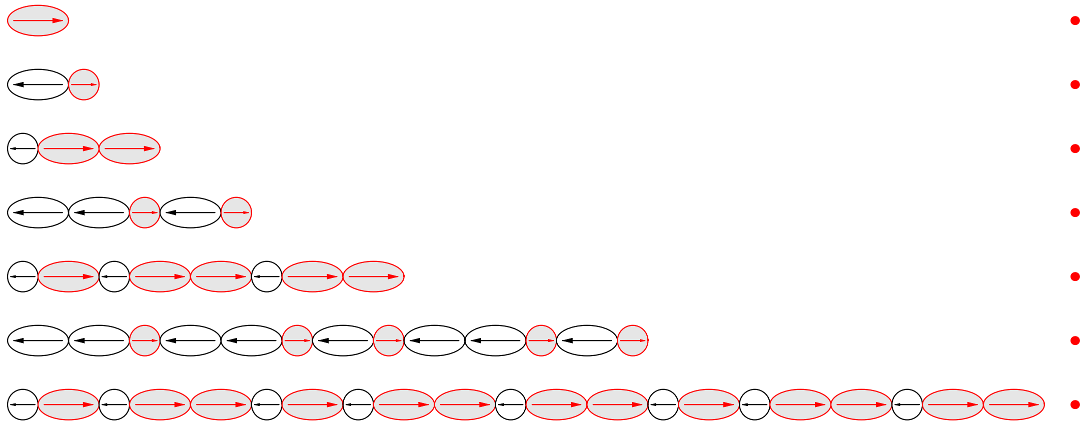


Figure 1: Simulated cell layer growth

Cells of types *a* and *b* are depicted as ellipses and smaller circles, respectively. Cell orientation is visualized through different greylevels as well as arrows pointing left (white) and right (grey).

A more elegant functional style formulation of an interpretation function using *Mathematica*'s pattern matching and upvalue definition capabilities is the following.

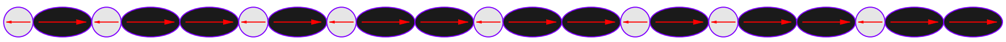
```
SetAttributes[CellLayerGraphics, {Listable}]

CellLayerGraphics[AXIOM[z__]] :=
  Fold[CellCircle[#2,#1]&,{{0,0},{}},{z}]

aL/: CellCircle[aL, {{x_,y_},g_List}] :=
       {{x+2,y},
        Append[g,ArrowedDisk[{x+1, y}, {1,0.5},
                   Direction -> Left]]}

aR/: CellCircle[aR, {{x_,y_},g_List}] :=
       {{x+2,y},
        Append[g,ArrowedDisk[{x+1, y}, {1,0.5},
                   HueValue -> 0.9,
                   Direction -> Right]]}

bL/: CellCircle[bL, {{x_,y_},g_List}] :=
       {{x+1,y},
        Append[g,ArrowedDisk[{x+.5, y}, {.5,0.5},
                   Direction -> Left]]}

bR/: CellCircle[bR, {{x_,y_},g_List}] :=
       {{x+1,y},
        Append[g,ArrowedDisk[{x+.5, y}, {.5,0.5},
                   HueValue -> 0.9,
                   Direction -> Right]]}
```

We define a listable function *CellLayerGraphics* which returns a Graphics object and the maximum x- and y-coordinate. The following command shows that the produced cell layer graphics extend horizontally up to a length of 34.

```
CellLayerGraphics[anabaenaGrowth] // Short
```

```
{{{2, 0}, {-Graphics-}}, {<<2>>}, <<4>>, {{34, 0}, <<1>>}}
```

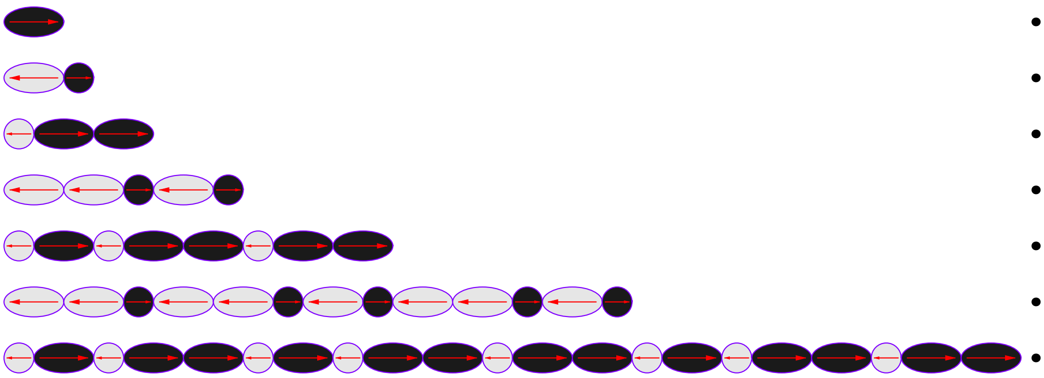
```
Show[% // Last // Last, AspectRatio -> Automatic];
```



For each of the cell symbols *aL*, *aR*, *bL*, and *bR* the upvalue *CellCircle[symbol,{pos,graphics}]* generates the appropriate graphics object at position *pos* and appends it to the *graphics* list. The following function *CellLayerPlot* extracts the graphics descriptions from the result produced by *CellLayerGraphics* and shows the graphics extended by the terminal point as described above.

```
CellLayerPlot[cells_List] :=
Module[{maxCoord,cellGraphs},

  cellGraphs = Map[CellLayerGraphics,cells];
  maxCoord = First @ Last @ cellGraphs;

  Map[
    Show[{# // Last,
          Graphics[Point[maxCoord+{.5,0}]]},
      PlotRange -> All,
      AspectRatio -> Automatic]&,
    cellGraphs
  ]
]
```

```
CellLayerPlot[anabaenaGrowth];
```



Context-sensitive parallel rewriting

Starting with a simple example ...

We use four simple rules where symbols *a*,*b* and *c* are only rewritten whenever they appear within certain contexts. For *a* and *b* only one context (left or right) is important whereas *c* is only replace by *d* if *c* appears within a (*c*,*d*)-context. We only give the formatted output of the rule system which is defined in the same notation as described above.

```
ILSystem1
```

```
Axiom: baaaaaaad

Rules: b < a >  -> b
        < b > b -> c
        < b > d -> c
       c < c > d -> d
```

These rules generate the following string sequence starting from the defined axiom:

```
anabaenaGrowth1 = runKLSystem[ILSystem1, 15]
```

```
{baaaaaaad, bbaaaaaad, cbbaaaaad, ccbbaaaad,
  cccbbaaad, ccccbbaad, cccccbbad, ccccccbbd,
  ccccccccd, cccccccdd, ccccccddd, cccccdddd,
  ccccddddd, cccdddddd, ccddddddd, cdddddddd}
```

In columnform the spreading of the *c* symbols through the string is more obvious:

```
ColumnForm[%]
```

```
baaaaaaad
bbaaaaaad
cbbaaaaad
ccbbaaaad
cccbbaaad
ccccbbaad
cccccbbad
ccccccbbd
cccccccccd
ccccccccdd
cccccccddd
ccccccdddd
cccccddddd
ccccdddddd
cccddddddd
ccdddddddd
cddddddddd
```

This can be interpreted as an example of cellular interaction, where a hormone (*c*) diffuses along a filament. However, there is at least one shortcoming with this rewrite system: the cell layer extension is predefined by the length of the axiom string. The rule system would become more flexible if we could parametrize the layer extension. We use this concept of parametrization in the next section.

Parametrized L-systems

Up to now we only used simple symbols as the rule system alphabet. If we parametrize these symbols - which means that we use expressions instead of symbols -we are able to control rule applications by environment variables.
 We introduce an environment variable *w* to control cell layer width

```
w = 11; (* cell layer width *)
```

and use this variable in the following parametrized L-system.

```
ILSystem2
```

```
Axiom: a[1]b

Rules: a[i_ /; i < w] < b >  -> a[1 + i]b
        < a[i_ /; i > 1] > c -> c
       a[15] < b >  -> c
       c < b >  ->
       a[i_ /; i > 1] < c >  ->
       a[1] < c >  -> b
```

```
anabaenaGrowth2 = runKLSystem[ILSystem2, 30];

ColumnForm[anabaenaGrowth2,Center]
```

```
                    a[1]b
                  a[1]a[2]b
                a[1]a[2]a[3]b
              a[1]a[2]a[3]a[4]b
            a[1]a[2]a[3]a[4]a[5]b
          a[1]a[2]a[3]a[4]a[5]a[6]b
        a[1]a[2]a[3]a[4]a[5]a[6]a[7]b
      a[1]a[2]a[3]a[4]a[5]a[6]a[7]a[8]b
    a[1]a[2]a[3]a[4]a[5]a[6]a[7]a[8]a[9]b
 a[1]a[2]a[3]a[4]a[5]a[6]a[7]a[8]a[9]a[10]b
a[1]a[2]a[3]a[4]a[5]a[6]a[7]a[8]a[9]a[10]a[11]b
a[1]a[2]a[3]a[4]a[5]a[6]a[7]a[8]a[9]a[10]a[11]c
 a[1]a[2]a[3]a[4]a[5]a[6]a[7]a[8]a[9]a[10]c
    a[1]a[2]a[3]a[4]a[5]a[6]a[7]a[8]a[9]c
      a[1]a[2]a[3]a[4]a[5]a[6]a[7]a[8]c
        a[1]a[2]a[3]a[4]a[5]a[6]a[7]c
          a[1]a[2]a[3]a[4]a[5]a[6]c
            a[1]a[2]a[3]a[4]a[5]c
              a[1]a[2]a[3]a[4]c
                a[1]a[2]a[3]c
                  a[1]a[2]c
                    a[1]c
                    a[1]b
                  a[1]a[2]b
                a[1]a[2]a[3]b
              a[1]a[2]a[3]a[4]b
            a[1]a[2]a[3]a[4]a[5]b
          a[1]a[2]a[3]a[4]a[5]a[6]b
        a[1]a[2]a[3]a[4]a[5]a[6]a[7]b
      a[1]a[2]a[3]a[4]a[5]a[6]a[7]a[8]b
    a[1]a[2]a[3]a[4]a[5]a[6]a[7]a[8]a[9]b
```

Anabaena catenula alga growth

We come back to our *Anabaena catenula* example. Now we want to control the cells' lifetime; at the end of their development time the cells of type *a* and *b* divide into two successor cells.

```
lifeTimeA = 3; (* development time cells a and b *)
lifeTimeB = 2;
```

These environment variables are used in the following L-system:

```
ILAxiom3 := AXIOM[ Al[1] ];

ILRules3 :=
  LRULES[
    LRule[LEFT[], PRED[ Al[lifeTimeA] ], RIGHT[],
                  SUCC[ Al[1],Br[1] ]
    ],
    LRule[LEFT[], PRED[ Ar[lifeTimeA] ], RIGHT[],
                  SUCC[ Bl[1],Ar[1] ]
    ],
    LRule[LEFT[], PRED[ Bl[lifeTimeB] ], RIGHT[],
                  SUCC[ Al[1],Br[1] ]
    ],
    LRule[LEFT[], PRED[ Br[lifeTimeB] ], RIGHT[],
                  SUCC[ Bl[1],Ar[1] ]
    ],
    LRule[LEFT[], PRED[ Al[i_ /; i < lifeTimeA] ],
          RIGHT[],
                  SUCC[ Al[i+1] ]
    ],
    LRule[LEFT[], PRED[ Ar[i_ /; i < lifeTimeA] ],
          RIGHT[],
                  SUCC[ Ar[i+1] ]
    ],
    LRule[LEFT[], PRED[ Bl[i_ /; i < lifeTimeB] ],
          RIGHT[],
                  SUCC[ Bl[i+1] ]
    ],
    LRule[LEFT[], PRED[ Br[i_ /; i < lifeTimeB] ],
          RIGHT[],
                  SUCC[ Br[i+1] ]
    ]
  ];
```

```
ILSystem3 := LSystem[ILAxiom3,ILRules3];

skipPattern = {Null};

postProcessingFunction := (Flatten[#,Infinity,List])&;
```

```
ILSystem3
```

```
Axiom: A [1]
        l

Rules:  < A [3] >  -> A [1]B [1]
           l           l    r
        < A [3] >  -> B [1]A [1]
           r           l    r
        < B [2] >  -> A [1]B [1]
           l           l    r
        < B [2] >  -> B [1]A [1]
           r           l    r
        < A [i_ /; i < lifeTimeA] >  -> A [1 + i]
           l                              l
        < A [i_ /; i < lifeTimeA] >  -> A [1 + i]
           r                              r
        < B [i_ /; i < lifeTimeB] >  -> B [1 + i]
           l                              l
        < B [i_ /; i < lifeTimeB] >  -> B [1 + i]
           r                              r
```

```
anabaenaGrowth3 = runKLSystem[ILSystem3, 9];

ColumnForm[anabaenaGrowth3]
```

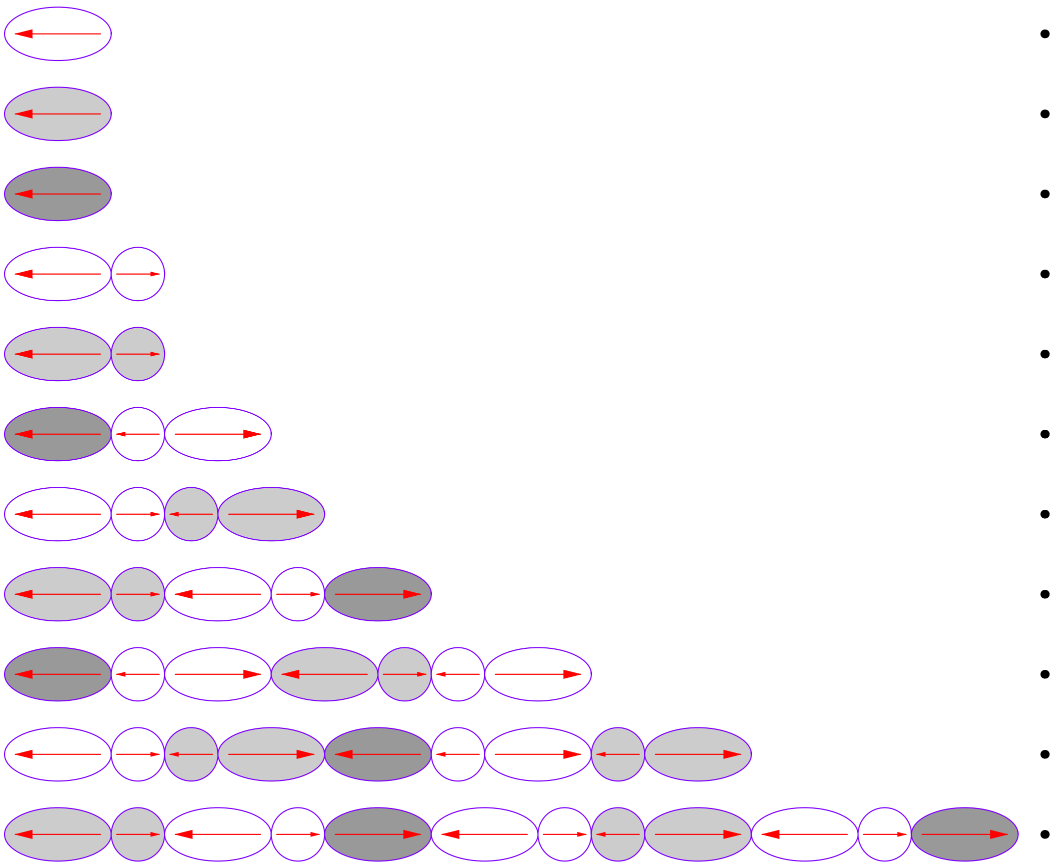
```
A [1]
 l
A [2]
 l
A [3]
 l
A [1]B [1]
 l    r
A [2]B [2]
 l    r
A [3]B [1]A [1]
 l    l    r
A [1]B [1]B [2]A [2]
 l    r    l    r
A [2]B [2]A [1]B [1]A [3]
 l    r    l    r    r
A [3]B [1]A [1]A [2]B [2]B [1]A [1]
 l    l    r    l    r    l    r
A [1]B [1]B [2]A [2]A [3]B [1]A [1]B [2]A [2]
 l    r    l    r    l    l    r    l    r
```

Graphical Interpretation

The interpretation function only has to be extended in order to take into account the lifetime of the two cell classes *a* and *b*. The developmental states of the cells are depicted by letting the cells become darker with growing age. The arrows signal cell orientation.

```
SetAttributes[CellLayerGraphics, {Listable}]

CellLayerGraphics[AXIOM[z__]] :=
  Fold[CellCircle[#2,#1]&,{{0,0},{}},{z}]

Al/: CellCircle[Al[i_], {{x_,y_},g_List}] :=
       {{x+2,y},
        Append[g,ArrowedDisk[{x+1, y}, {1,0.5},
                   HueValue -> 0.2 (i-1),
                   Direction -> Left]]}

Ar/: CellCircle[Ar[i_], {{x_,y_},g_List}] :=
       {{x+2,y},
        Append[g,ArrowedDisk[{x+1, y}, {1,0.5},
                   HueValue -> 0.2 (i-1),
                   Direction -> Right]]}

Bl/: CellCircle[Bl[i_], {{x_,y_},g_List}] :=
       {{x+1,y},
        Append[g,ArrowedDisk[{x+.5, y}, {.5,0.5},
                   HueValue -> 0.2 (i-1),
                   Direction -> Left]]}

Br/: CellCircle[Br[i_], {{x_,y_},g_List}] :=
       {{x+1,y},
        Append[g,ArrowedDisk[{x+.5, y}, {.5,0.5},
                   HueValue -> 0.2 (i-1),
                   Direction -> Right]]}
```

```
devTimeA = 3;
devTimeB = 2;

anabaenaGrowth3 = runKLSystem[ILSystem3, 10];

CellLayerPlot[anabaenaGrowth3];
```



```
Figure 2: Algae growth with two classes of cells a (big)
          and b (small).
          Development times are 3 and 2, respectively.
```

As in the examples above the *a*-cells are the bigger ones. Each *a*-cell needs three timesteps to mature. Then the cell divides into a new *a*-cell (with same orientation) and a smaller *b*-cell with opposite orientation. After two timesteps the *b*-cell is ripe for dividing into a *b*-cell with left orientation and an *a*-cell oriented to the right. It is an interesting exercise to think about resulting cell layer growthrates for different lifetimes of the cells (see the following figure).

```
devTimeA = 2;
devTimeB = 5;

anabaenaGrowth4 = runKLSystem[ILSystem3, 15];

CellLayerPlot[anabaenaGrowth4];
```

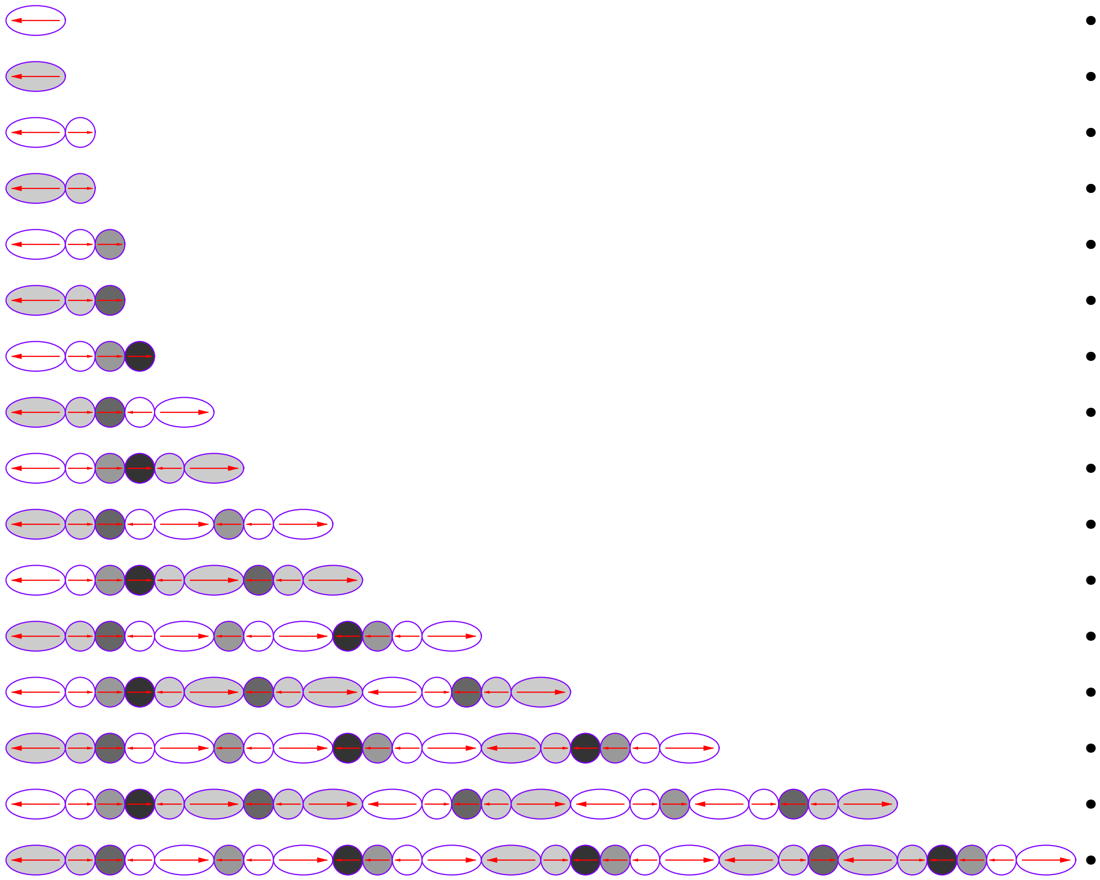


Figure 3: Algae growth with lifetimes 2 and 5 for cells *a* and *b*, respectively

Conclusion

We have briefly shown a simple implementation of parallel rewrite systems with some example applications of growing cell layers. In the next sections we will show how L-systems are applied for an implicit description of fractals and we will describe concepts for graphical interpretation in 2- and 3-dimensional space.

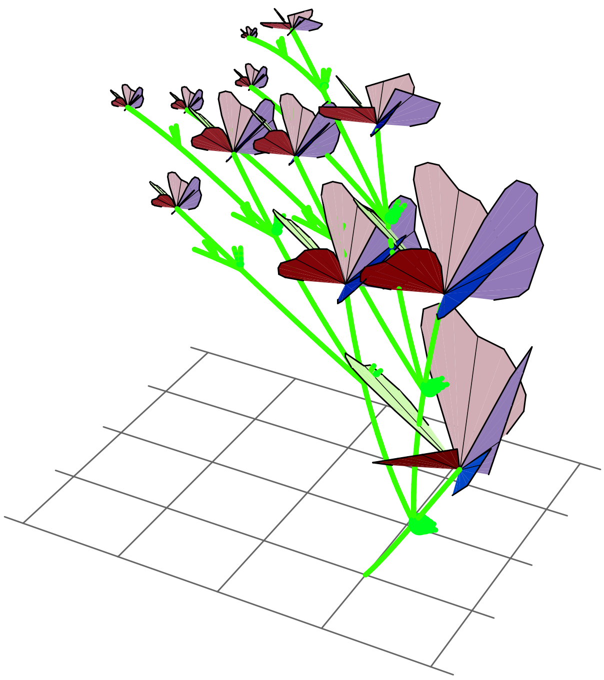


Figure 4: Example of an artificial flower the growth of which is described by an L-system.

References

[Bac59] Backus, J.W., *The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM conference*, in: Proc. Intl. Conf. on Information Processing, 125-132, UNESCO, 1959.

[Cho56] Chomsky, N., *Three models for the description of language*, IRE Transactions on Information Theory, 2(3):113-124, 1956.

[Lin68] Lindenmayer, A., *Mathematical models for cellular interaction in development, Parts I and II*, Journal of Theoretical Biology, 18:280-315, 1968.

[Pru89] Prusinkiewicz, P., and Hanan, J., *Lindenmayer Systems, Fractals, and Plants*, Springer-Verlag, New York, 1989.

[Pru90] Prusinkiewicz, P., and Lindenmayer, A., *The Algorithmic Beauty of Plants*, Springer-Verlag, New York, 1990.

About the author

Christian Jacob
Chair of Programming Languages
University of Erlangen-Nˆrnberg
Martensstr. 3
D-91058 Erlangen, GERMANY
jacob@informatik.uni-erlangen.de