L-Systems

Simulation of development and growth
The algorithmic beauty of plants
L-Systems

- The central concept of L-Systems is that of rewriting
- A classical example of an object defined using rewriting rules is the von Koch snowflake
- Mandelbrot states it as a rewriting system
  - One begins with *two shapes*, an *initiator* and a *generator*.
  - The latter is an oriented broken line made up of $N$ equal sides of length $r$.
  - Thus each stage of the construction begins with a broken line and consists in *replacing* each straight interval with a copy of the *generator*, reduced and displaced so as to have the same end points as those of the interval being replaced.

Iterations of the von Koch snowflake

initiator

generator
Other rewriting systems

- Rewriting systems on graphs
- Rewriting systems on rectangular arrays and matrices (→ Cellular automata)
- *Rewriting systems on character strings*
History of string rewriting

- Begin 19th century: Thue provided first formal definition of a string rewriting system (srw)
- Late 50ties: Chomsky investigated syntactical features of natural languages using srw
- 1960: Backus and Naur used rewriting notation in the definition of programming language ALGOL
- 1952: Equivalence between Backus Naur form and context free class of Chomsky’s grammar (ccg) was realized
- 1968 the biologist A. Lindenmayer introduced L-Systems to model multicellular plant growth
- As opposed to ccg rewriting rules are applied to all letters in the string simultaneously; there are languages that can be expressed in L-Systems but not in ccg

Relation between Chomsky classes and L-systems

- Relations between Chomsky classes of languages and language classes generated by L-systems.
- The symbols OL and IL denote language classes generated by context-free and context-sensitive L-systems.
DOL-systems

- simplest class of L-systems, those which are deterministic and context-free, called DOL-systems.
- consider strings (words) built of two letters ‘a’ and ‘b’, which may occur many times in a string.
- Each letter is associated with a rewriting rule.
- The rule ‘a → ab’ means that the letter ‘a’ is to be replaced by the string ‘ab’, and the rule ‘b → a’ means that the letter ‘b’ is to be replaced by ‘a’.
- The rewriting process starts from a distinguished string called the axiom.

Letters are simultaneously replaced in an derivation step

→ first five derivations of the described DOL-system with axiom ‘b’
Formal definition of a DOL-System

Let $V$ denote an alphabet, $V^*$ the set of all words over $V$, and $V^+$ the set of all nonempty words over $V$. A string OL-system is an ordered triplet $G = (V, \omega, P)$ where $V$ is the alphabet of the system,

- $\omega \in V^+$ is a nonempty word called the axiom and $P \subseteq V \times V^*$ is a finite set of productions.

- A production $(a, \chi) \in P$ is written as $a \rightarrow \chi$. The letter $a$ and the word $\chi$ are called the predecessor and the successor of this production, resp.

- It is assumed that for any letter $a \in V$, there is exactly one word $\chi \in V^*$ such that $a \rightarrow \chi$. (if not we assume $a \rightarrow a$)

*if we say 'at least one word' it is a OL system, but not a deterministic one DOL.
Derivation

- Let $\mu = a_1 \ldots a_m$ be an arbitrary word over $V$.
- The word $\nu = \chi_1 \ldots \chi_m \in V^*$ is directly derived from (or generated by) $\mu$, noted $\mu \Rightarrow \nu$, if and only if $a_i \rightarrow \chi_i$ for all $i = 1, \ldots, m$.
- A word $\nu$ is generated by $G$ in a derivation of length $n$ if there exists a developmental sequence of words $\mu_0, \mu_1, \ldots, \mu_n$ such that $\mu_0 = \omega, \mu_n = \nu$ and $\mu_0 \Rightarrow \mu_1 \Rightarrow \ldots \Rightarrow \mu_n$. 

Diagram:

```
  b
 / \
|   |
a
 / \
|   |
ab
 / \
|   |
aba
 / \
|   |  \
aba
 / \
|   |
aba
 / \
|   |
aba
 / \
|   |
aba
 / \
|   |
aba
 / \
|   |
aba
 axiom $\omega$
```

devolopmental sequence

development from a
Example: Development of a filament of the bacteria *Anabaena catenula*

- The symbols $a$ and $b$ represent cytological states of the cells.
- The subscripts $l$ and $r$ indicate cell polarity, specifying the positions in which daughter cells of type $a$ and $b$ will be produced.
- L-system describes development:
  - $\omega : a_r$
  - $p1 : a_r \rightarrow a_l b_r$
  - $p2 : a_l \rightarrow b_r a_r$
  - $p3 : b_r \rightarrow a_r$
  - $p4 : b_l \rightarrow a_l$

  $a_r$, $a_l b_r$, $b_l a_r a_l b_r$, $b_l a_r a_l b_r a_r a_l a_r a_r \ldots$
Model validation

- Under a microscope, the filaments appear as a sequence of cylinders of various lengths, with $a$-type cells longer than $b$-type cells.
- Letters of the L-system alphabet are represented graphically as shorter or longer rectangles with rounded corners.
- Generated structures are one-dimensional chains of rectangles.
- Schematic image of filament development resembles observed multi-cellular patterns that can be compared to observed patterns.
- Intermediate continuous cell growth is not modeled.
More sophisticated graphical representations

- For modelling phenomena such as branching in plants of higher order we need more sophisticated graphical representations.
- The first results in this direction were published in 1974 by Frijters and Lindenmayer, and Hogeweg and Hesper.
- Work on relating L-systems to fractals, space filling curves, and Indian art kolam patterns and music followed*
- Prusinkiewicz focused on an interpretation based on a LOGO-style turtle


P. Prusinkiewicz (1987). Applications of L-systems to computer imagery. In H. Ehrig, M. Nagl, A. Rosenfeld, and G. Rozenberg, editors, Graph grammars and their application to computer science; LNCS 291
The basic idea of turtle interpretation

- A state of the turtle is defined as a triplet \((x, y, \alpha)\)
- the Cartesian coordinates \((x, y)\) represent the turtle’s position, and the angle \(\alpha\), called the heading, is interpreted as the direction in which the turtle is facing.
- Given the step size \(d\) and the angle increment \(\delta\), the turtle can respond commands represented by the following symbols

- **F** Move forward a step of length \(d\). The state of the turtle changes to \((x, y, \alpha)\), where \(x = x + d \cos \alpha\) and \(y = y + d \sin \alpha\). A line segment between points \((x, y)\) and \((x, y)\) is drawn.
- **f** Move forward a step of length \(d\) without drawing a line.
- **+** Turn left by angle \(\delta\). The next state of the turtle is \((x, y, \alpha + \delta)\). The positive orientation of angles is counterclockwise.
- **−** Turn right by angle \(\delta\). The next state of the turtle is \((x, y, \alpha - \delta)\).
Example for turtle graphics interpretation

(a) F

(b) FFF-FF-F-F+F+FF-F-FFF

Start
This method can be applied to interpret strings which are generated by L-systems.

Approximations of the quadratic Koch island taken from Mandelbrot’s book [95, page 51].

The initiator corresponds to the axiom and the generator corresponds to the production successor.

L-systems specified in this way can be perceived as codings for Koch constructions.

\[ \omega : F \rightarrow F - F - F - F \]
\[ p : F \rightarrow F - F + F + FF - F - F + F \]

\( \delta = 90^\circ \), d is decreased 4 times per derivation, n = number of derivations
More L-systems coding fractals with initiator and generator

\begin{align*}
\text{a:} & \quad n = 2, \quad \delta = 90^\circ \\
& \quad F \rightarrow F-F-F-F \\
& \quad F \rightarrow F+FF-FF-F-F+F+F+F \\
& \quad F-F-F+F+FF+FF-F \\
\text{b:} & \quad n = 4, \quad \delta = 90^\circ \\
& \quad -F \\
& \quad F \rightarrow F+F-F-F+F+F+F+F-F \\
\end{align*}
Combinations of islands and lakes

- A complication occurs when the curve is not connected
- A second production rule with predecessor is then required

\[ n = 2, \quad \delta = 90^\circ \]
\[ F+F+F+F \]
\[ F \rightarrow F+f-FF+F+FF+Ff+FF-f+FF-F+FF-Ff+FF \]
\[ f \rightarrow fffffff \]
More L-systems coding fractals (1)

- The ease of modifying L-systems makes them suitable for developing new von Koch curves.
- Try to gradually develop by inserting, deleting, replacing symbols!
- A sequence of Koch curves obtained by successive modification of the production successor
- Similar L-systems can give rise to very dissimilar graphical representations
L-system synthesis

- we often wish to construct an L-system which captures a given structure or sequence of structures representing a developmental process.
- This is called the *inference problem* in the theory of L-systems.
- Although some algorithms for solving it were reported in the literature*, they are still too limited to be of practical value in the modeling of higher plants.
- Edge rewriting and node rewriting are more intuitive approaches to this problem.

Edge rewriting and node rewriting

- Terminology is borrowed from graph theory
- In the case of edge rewriting, productions substitute figures for polygon edges
- In node rewriting, productions operate on polygon vertices (in L-systems symbols are needed for them)
- Both techniques capture the recursive structure of the figures
- The concepts are illustrated using abstract curves, they apply to branching structures found in plants as well.
Edge rewriting

- Edge rewriting can be viewed as an extension of Koch constructions.
- The figure shows the *dragon curve* and the L-system that generated it.
- Both the $F_l$ and $F_r$ symbols represent edges created by the turtle executing the “move forward” command.
- The productions substitute $F_l$ or $F_r$ edges by pairs of lines forming left or right turns.
- Many interesting curves can be obtained assuming two types of edges, “left” and “right.”
- Dragon curve can be also obtained by paper-folding and looks similar to Julia set.

\[
\begin{align*}
F_l &
\rightarrow F_l + F_r + \\
F_r &
\rightarrow -F_l - F_r
\end{align*}
\]

\[a \ n = 10, \ \delta = 90^\circ\]
Example

- Synthesis of an Sierpinski gasket using left turn right turn system
- Edge rewriting makes synthesis more intuitive

\[
\begin{align*}
\theta &= 60^\circ \\
F_r &\rightarrow F_r + F_1 + F_r \\
F_r &\rightarrow F_1 - F_r - F_1
\end{align*}
\]
FASS curve construction

F: space filling
A: self-avoiding
S: simple
S: self-similar

Hexagonal Gosper-curve

a
n=4, $\delta=60^\circ$

$F_1$

$F_1 \rightarrow F_1+F_x++F_x-F_1--F_1F_1-F_x+$

$F_x \rightarrow -F_1+F_xF_x++F_x+F_1--F_1-F_x$

Quadratic Gosper or E-curve

b

n=2, $\delta=90^\circ$

$-F_x$

$F_1 \rightarrow F_1F_1-F_x-F_x+F_1+F_1-F_xF_1+$

$F_x+F_1F_1F_x-F_1+F_x+F_1F_1+$

$F_x-F_1F_x-F_1+F_1+F_1F_x-$

$F_x \rightarrow +F_1F_1-F_x-F_x+F_1+F_1F_x+F_1-$

$F_xF_x-F_1+F_1F_xF_x-F_1-$

$F_x+F_1+F_x-F_x-F_1+F_1+F_xF_x$
FASS curves

- can be thought of as finite, self-avoiding approximations of curves that pass through all points of a square (space-filling curves).
- McKenna presented an algorithm for constructing FASS curves using edge rewriting.
- It exploits the relationship between such a curve and a recursive subdivision of a square into tiles.
- Later we will see an node-replacement algorithm for FASS curves.
Why is the E-Curve space filling?

- The polygon replacing an edge $F_l(a)$ approximately fills the square on the left side of $F_l(b)$.
- Similarly, the polygon replacing an edge $F_r(c)$ approximately fills the square on the right side of that edge (d).
- In the next derivation step, each of the 25 tiles associated with the curves (b) or (d) will be covered by their reduced copies.
- A recursive application of this argument indicates that the whole curve is approximately space-filling.

Left and right edges are distinguished by the direction of ticks.

\[
F_1 \rightarrow F_1 F_1 + F_r + F_l - F_1 - F_1 + F_r + F_l F_1 - F_r - F_1 F_1 F_r + \\
F_1 - F_r - F_1 F_1 - F_r + F_1 F_1 + F_r + F_1 - F_1 - F_r F_r + \\
F_r \rightarrow - F_1 F_1 + F_r + F_l - F_1 - F_1 F_r - F_1 F_r + F_l F_r + F_1 + F_r - \\
F_1 F_r F_r + F_1 + F_r F_1 - F_r + F_r + F_1 - F_1 - F_r F_r
\]
Why is the E-curve self-avoiding

- the generating polygon is self-avoiding, and
- no matter what the relative orientation of the polygons lying on two adjacent tiles, their union is a self-avoiding curve.
- The first property is obvious, while the second can be verified by considering all possible relative positions of a pair of adjacent tiles:
Optimality of the E-curve

- The simplicity of a generating polygon can be measured by the number of edges in the production.
- Using a computer program to search the space of generating polygons, McKenna found that the E-curve is the simplest FASS curve obtained by edge replacement in a square grid.
- The relationship between edge rewriting and tiling of the plane extends to branching structures, providing a method for constructing and analyzing L-systems which operate according to the edge-rewriting paradigm.
The idea of node rewriting is to substitute new polygons for nodes of the Subfigures predecessor curve.

Turtle interpretation is extended by symbols which represent arbitrary subfigures.

Each subfigure $A$ from a set of subfigures $\mathcal{A}$ is represented by:

- two contact points, called entry point $P_A$ and exit point $Q_A$, and
- two direction vectors, called entry vector $p_A$ and exit vector $q_A$.

During turtle interpretation of a string $\nu$, a symbol $A \in \mathcal{A}$ incorporates the corresponding subfigure into a picture.

To this end $A$ is translated and rotated in order to align its entry point $P_A$ and direction $p_A$ with the current position and orientation of the turtle.

Having placed $A$, the turtle is assigned the resulting exit position $Q_A$ and direction $q_A$. 
Node rewriting - example

- For example, assuming that the contact points and directions of Recursive subfigures $L_n$ and $R_n$ are as in the figure and figures $L_{n+1}$ and $R_{n+1}$ formulas are captured by the following formulas:

  \[ L_{n+1} = +R_n F - L_n F L_n - F R_n \]
  \[ R_{n+1} = -L_n F + R_n F R_n + F L_n \]

- Suppose that curves $L_0$ and $R_0$ are given.
Node rewriting and recursion

- One way of evaluating the string $L_n$ (or $R_n$) for $n > 0$ is to generate successive strings recursively, in the order of decreasing value of index $n$. (see formula below)
- The generation of string $L_n$ can be considered as a string-rewriting mechanism, where the symbols on the left side of the recursive formulas are substituted by corresponding strings on the right side.
- The substitution proceeds in a parallel way with $F$, $+$ and $-$ replacing themselves.
- Since all indices in any given string have the same value, they can be dropped, provided that a global count of derivation steps is kept.
- Consequently, string $L_n$ can be obtained in a derivation of length $n$ using the following L-system:

$$L_{n+1} = +R_n F - L_n F L_n - F R_n +$$
$$R_{n+1} = -L_n F + R_n F R_n + F L_n -$$

$$L_2 = +R_1 F - L_1 F L_1 - F R_1 +$$
$$= +(-L_0 F + R_0 F R_0 + F L_0 -) F - (+R_0 F - L_0 F L_0 - F R_0 +) F(-L_0 F + R_0 F R_0 + F L_0 -) +$$

$$\omega : L$$
$$p_1 : L \rightarrow +RF - LFL - FR +$$
$$p_2 : R \rightarrow -LF + RFR + FL -$$
Pure curves and subfigures

\[
\begin{align*}
L_{n+1} &= +R_nF - L_nFL_n - FR_n + \\
R_{n+1} &= -L_nF + R_nFR_n + FL_n -
\end{align*}
\]

\[
\omega : \quad L \\
p_1 : \quad L \rightarrow +RF - LFL - FR + \\
p_2 : \quad R \rightarrow -LF + RFR + FL -
\]

- In order to further reduce the node rewriting system to an L system we have to replace L and R by subfigures.
- In the ideal case the subfigures reduce to single points (pure curves), and are replaced by empty string.
- Alternatively, they can be left in the string and ignored by the turtle during string interpretation.
- As in the case of edge rewriting, the relationship between node rewriting and tilings of the plane extends to branching structures (see next part).
- It offers a method for synthesizing L-systems that generate objects with a given recursive structure, and links methods for plant generation based on L-systems with those using iterated function systems.
Consider an array of $m \times m$ square tiles, each including a smaller square, called a \textit{frame}. Each frame bounds an open self-avoiding polygon. The endpoints of this polygon coincide with the two contact vertices of the frame. Suppose that a single-stroke line running through all tiles can be constructed by connecting the contact vertices of neighboring frames using short horizontal or vertical line segments. A FASS curve can be constructed by the recursive repetition of this connecting pattern.

To this end, the array of $m \times m$ connected tiles is considered a \textit{macrotile} which contains an open polygon inscribed into a \textit{macroframe}. An array of $m \times m$ macrotiles is formed, and the polygons inscribed into the macroframes are connected together. This construction is carried out recursively, with $m \times m$ macrotiles at level $n$ yielding one macrotile at level $n + 1$. 
Construction of FASS curves

Sample FASS curve constructed using tiles with contact points positioned along a tile edge using a 3x3 macrotile

\[ a \quad n=2, \quad \delta=90^\circ \]

L
L→LFRFL→F→RFLFR+F+LFRFL
R→RFLFR+F+LFRFL→F→RFLFR
The classes of curves that can be generated using the edge-rewriting and node-rewriting techniques are not disjoint.

We illustrate this by means of an example.

Reconsider the L-system that generates the dragon curve using edge replacement:

- Consider a production can have a string consisting of more than one predecessor.
- (such systems are called pseudo L-systems)
- the symbols $l$ and $r$ are not interpreted by the turtle.

$$
\begin{align*}
\omega : & \quad Fl \\
p_1 : & \quad Fl \rightarrow Fl + Fr + \\
p_2 : & \quad Fr \rightarrow -Fl - Fr
\end{align*}
$$

- Production $p_1$ replaces the letter $l$ by the string $l + rF$ while the leading letter $F$ is left intact.
In practice, the choice between edge rewriting and node rewriting is often a matter of convenience.

Neither approach offers an automatic, general method for constructing L-systems that capture given structures.

The distinction between edge and node rewriting makes it easier to understand the intricacies of L-system operation.

Specifically, the problem of filling a region by a self-avoiding curve is biologically relevant, since some plant structures, such as leaves, may tend to fill a plane without overlapping.
3-D L-Systems
3-D L-systems

- Turtle interpretation of L-systems can be extended to three dimensions following the ideas of Abelson and diSessa.
- The key concept is to represent the current orientation of the turtle in space by three vectors $H, L, U$, indicating the turtle’s heading, the direction to the left, and the direction up.
- These vectors have unit length, are perpendicular to each and satisfy the equation $H \times L = U$.
- Rotations of the turtle are then expressed by the equation
  $$[H' \ L' \ U'] \leftarrow [H \ L \ U] R$$
  with rotation matrix $R$

Rotation matrix $R$

$$R_U(\alpha) = \begin{bmatrix} \cos \alpha & \sin \alpha & 0 \\ -\sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_L(\alpha) = \begin{bmatrix} \cos \alpha & 0 & -\sin \alpha \\ 0 & 1 & 0 \\ \sin \alpha & 0 & \cos \alpha \end{bmatrix}$$

$$R_H(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{bmatrix}$$
3-D L-system

\[ n=2, \quad \delta=90^\circ \]

\[ A \quad \rightarrow \quad B-F+CFC+F-D\&F\wedge D-F+\&\&CFC+F+B/\]

\[ B \quad \rightarrow \quad A\&F\wedge CFB\wedge F\wedge D\wedge\neg F-D\wedge F\wedge B\wedge FC\wedge F\wedge A// \]

\[ C \quad \rightarrow \quad D\wedge F\wedge B-F+C\wedge F\wedge A\&\&FA\&F\wedge C+F+B\wedge F\wedge D// \]

\[ D \quad \rightarrow \quad CFB-F+B\wedge FA\&F\wedge A\&\&FB-F+B\wedge FC// \]

+ turn left by angle \( \delta \), using rotation matrix \( R_U(\delta) \)
- turn right by angle \( \delta \), using rotation matrix \( R_U(-\delta) \).
& Pitch down by angle \( \delta \), using rotation matrix \( R_L(\delta) \).
^ Pitch up by angle \( \delta \), using rotation matrix \( R_L(-\delta) \).
\ Roll left by angle \( \delta \), using rotation matrix \( R_H(\delta) \).
/ Roll right by angle \( \delta \), using rotation matrix \( R_H(-\delta) \).
| Turn around, using rotation matrix \( R_U(180^\circ) \).

three-dimensional extension of the Hilbert curve. Colors represent three-dimensional “frames” associated with symbols A (red), B (blue), C (green) and D (yellow).
Branching systems

- According to the rules presented so far, the turtle interprets a character string as a sequence of line segments.
- The edge sequences form paths from a distinguished node, called the *root* or *base*, to the *terminal nodes*.
- In the biological context, these edges are referred to as *branch segments*.
- A segment followed by at least one more segment in some path is called an *internode*.
- A terminal segment (with no succeeding edges) is called an *apex*. 
Axial trees

- An axial tree is a special type of rooted tree.
- At each of its nodes, it has most one outgoing straight segment.
- All remaining edges are called lateral or side segments.
- A sequence of segments is called an axis, if:
  - the first segment in the sequence originates at the root of the tree or as a lateral segment at some node.
  - each subsequent segment is a straight segment.
  - the last segment is not followed by any straight segment in the tree.
Branches of axial trees

- Together with all its descendants, an axis constitutes a *branch*.
- A branch itself is an axial tree
- Axes and branches are ordered.
  - The axis originating at the root of the entire plant has order zero.
  - An axis originating as a lateral segment of an *n*-order parent axis has order *n+1*.
  - The order of a branch is equal to the order of its lowest-order or *main* axis.
Axial trees as topological objects

- Axial trees are purely topological objects.
- The geometric connotation of such terms as straight segment, lateral segment and axis should be viewed at this point as an intuitive link between the graph-theoretic formalism and real plant structures.
- The proposed scheme for ordering branches in axial trees was introduced originally by Gravelius.
- MacDonald [94, pages 110–121] surveys this and other methods applicable to biological and geographical data such as stream networks.

Horton and Strahler analysis

- Of special interest are methods proposed by Horton and Strahler, which served as a basis for synthesizing botanical trees.
- The left figure displays a tree obtained using Horton and Strahler analysis of branching patterns.

In order to model development of branching structures, a rewriting mechanism can be used that operates directly on axial trees. A rewriting rule, or *tree production*, replaces a predecessor edge by a successor axial tree in such a way that the starting node of the predecessor is identified with the successor’s base and the ending node is identified with the successor’s top. A *tree OL-system* $G$ is specified by three components:

- a set of edge labels $V$
- an initial tree $\omega$ with labels from $V$
- a set of tree productions $P$

Given the L-system $G$, an axial tree $T_2$ is directly derived from a tree $T_1$, noted $T_1 \Rightarrow T_2$, if $T_2$ is obtained from $T_1$ by simultaneous replacing each edge in $T_1$ by its successor according to the production set $P$. A tree $T$ is generated by $G$ in a derivation of length $n$ if there exists a sequence of trees $T_0, T_1, \ldots, T_n$ such that $T_0 = \omega$, $T_n = T$ and $T_0 \Rightarrow T_1 \Rightarrow \ldots \Rightarrow T_n$. 
Axial tree production systems

production rule

is transformed to

production rule applied
Bracketed 0L Systems

- The definition of tree L-systems does not specify the data structure for representing axial trees.
- One possibility is to use a list representation with a tree topology (initiator, generator).
- Alternatively, axial trees can be represented using *strings with brackets* (e.g. in bracketed OL-systems).
- New symbols ‘[‘ and ‘]’ are introduced to delimit a branch.
- They are interpreted by the turtle as follows:
  [  Push the current state of the turtle onto a pushdown operations stack.
  The information saved on the stack contains the turtle’s position and orientation, and possibly other attributes.
  ]  Pop a state from the stack and make it the current state of the turtle. No line is drawn, although in general the position of the turtle changes.

Example

Bracketed OL-systems

- Derivations in bracketed OL-systems proceed as in OL-systems without brackets.
- The brackets replace themselves.
- Examples of two-dimensional branching structures generated by bracketed OL-systems are shown in the figure.
Edge rewriting bracketed L systems

a  
\[ n=5, \delta=25.7^\circ \] 
F  
F \rightarrow F [+F] F [-F] F

b  
\[ n=5, \delta=20^\circ \] 
F  
F \rightarrow F [+F] F [-F] [F]

c  
\[ n=4, \delta=22.5^\circ \] 
F  
F \rightarrow FF [-F+F+F] + 
[+F-F-F]
Node rewriting bracketed L-systems

d
\[ n=7, \delta=20^\circ \]
X
X \rightarrow F [+X] F [-X] + X
F \rightarrow FF

e
\[ n=7, \delta=25.7^\circ \]
X
X \rightarrow F [+X] [-X] FX
F \rightarrow FF

f
\[ n=5, \delta=22.5^\circ \]
X
X \rightarrow F [- [X] + X] + F [+FX] - X
F \rightarrow FF
3-D Bush-like structure

- \( p_1 \) creates three new branches from an apex of the old branch.
- A branch consists of an edge \( F \) forming the initial internode, a leaf \( L \) and an apex \( A \) (which will subsequently create three new branches).
- \( p_2 \) and \( p_3 \) specify internode growth
- In subsequent derivation internode steps get longer and acquire new leaves
- Production \( p_4 \) specifies the leaf as a filled polygon with six edges. Its boundary is formed from the edges \( f \) enclosed between the braces \( \{ \) and \( \} \)
- The symbols \( ! \) and \( \& \) are used to decrement the diameter of segments and increment the current index to the color table, respectively.

\[
\begin{align*}
\omega & : A \\
\rho_1 & : A \rightarrow [\&FL!A]\\
\rho_2 & : F \rightarrow S F \quad \rho_3 : S \rightarrow F L \\
\rho_4 & : L \rightarrow [\&\&\&\&\&{-f+f+f-{-f+f+f}]
\end{align*}
\]
A flowering plant generated by an L-system

\( n=5 \quad \delta=18^\circ \)

\( \omega : \) plant

\( p_1 : \) plant \( \rightarrow \) internode + [ plant + flower ] -- //  
             [ -- leaf ] internode [ ++ leaf ] -- 
             [ plant flower ] ++ plant flower

\( p_2 : \) internode \( \rightarrow \) F seg [ // & & leaf ] [ // & & leaf ] F seg

\( p_3 : \) seg \( \rightarrow \) seg F seg

\( p_4 : \) leaf \( \rightarrow \) [‘ { +f–ff–f+ | +f–ff–f } ]

\( p_5 : \) flower \( \rightarrow \) [ & & & pedicel ‘ / wedge ///// wedge ////// wedge ////// wedge ////// wedge ]

\( p_6 : \) pedicel \( \rightarrow \) FF

\( p_7 : \) wedge \( \rightarrow \) [‘ & F ] [ { & & & & ef | ef } ]

- Note that the plant was generated in 5 iterations.
- Consider, the amount of information needed for directly describing the plant.
Summary L-Systems

- L-Systems are based on string rewriting using production rules (→ graph grammars)
- Turtle graphics give them a graphical interpretation
- L-Systems serve as models of development in biology, but also in other areas.
- Small changes of rules have often surprisingly large impact.
- Branching structures can be implemented using stacks
- Axial trees are important branching structures in nature (rivers, botanical trees, …)
- Edge and node rewriting in 3-D bracketed L-Systems can model complex self-similar objects such as flowering plants and bushes in a extremely compact way.
Outlook

- Realistic simulation systems of growing systems can consider production rules as used in L-Systems.
- L-Systems show the deep connection between language, dynamical systems, and graphical/real-world objects.
- L-Systems are still a relatively young research discipline, and for the future many applications of it are expected, not at least in systems simulation.