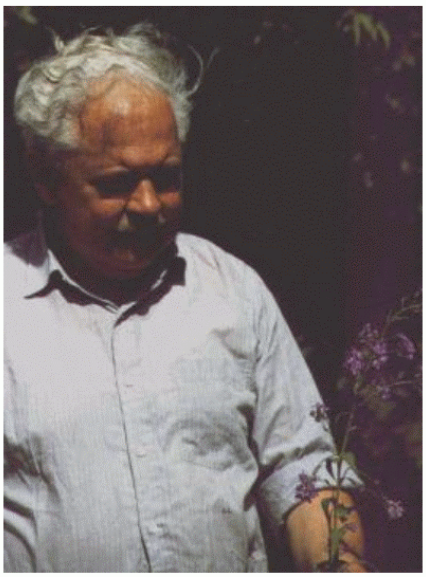
L-Systeme (Lindenmayer-Systeme)

Modellierung von Pflanzenarchitekturen
als Regelanwendungen: L-Systeme



Aristid Lindenmayer
(1925 – 1989)

im Vergleich zu den bisher vorgestellten Ansätzen der Morphogenese-Modellierung sind L-System-Modelle:

- *strukturorientiert*
- diskret in Zeit und Struktur (aber kontinuierlich im Raum: Lage, Richtung – Vorteil gegenüber CA)
- Topologie linear oder verzweigt
- feste Nachbarschaften

Grundlagen der L-Systeme

- Modellierung der Pflanzenmorphologie aus diskreten Einheiten
- Produktionsregeln bestimmen vom Startzustand iterativ das Pflanzenwachstum
- Selbstähnlichkeit „eingebaut", Fraktale als Grenzfall

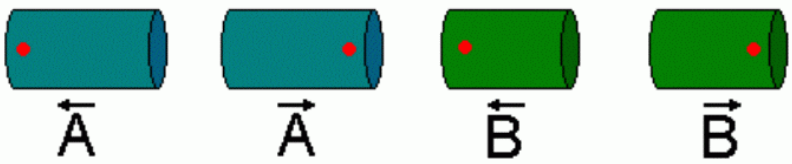
- Verwendung hauptsächlich für Vegetationsmodelle
  aber auch: Webmuster, Gebäude, Roboter, Tiere (Vermehrung, Nahrungsaufnahme), Melodien
- 2D- und 3D-Varianten
- Emulation von IFS möglich
- volle Mächtigkeit einer Programmiersprache
- *dynamische* Simulationen ($\Rightarrow$ Möglichkeit der Animation)
- Anbindung physikalisch oder biologisch begründeter Simulationsmodelle möglich (Kombination von Modell-Ansätzen)

Formalismus der L-Systeme:
aus der Theorie formaler Grammatiken

analog zu Chomsky-Grammatiken (regulär, kontextfrei, kontext-sensitiv etc.)
aber: in jedem Ableitungsschritt *parallele* Ersetzung aller Zeichen, auf die eine Regel anwendbar ist

von Aristid Lindenmayer (Botaniker) 1968 zur Modellierung des Wachstums von fadenförmigen Algen eingeführt (Verwandt-schaft zum CA-Ansatz)

Erstes Beispiel: Anabaena Catenula



Blaugrünes Bakterium, existiert in vier Varianten:

- „groß": Rechts-A und Links-A
- „klein": Rechts-B und Links-B

Beobachtungen:

- R-A teilt sich in L-A und R-B und R / L vertauscht
- R-B wird spontan zu R-A, L-B zu L-A

L-System für Anabaena Catenula

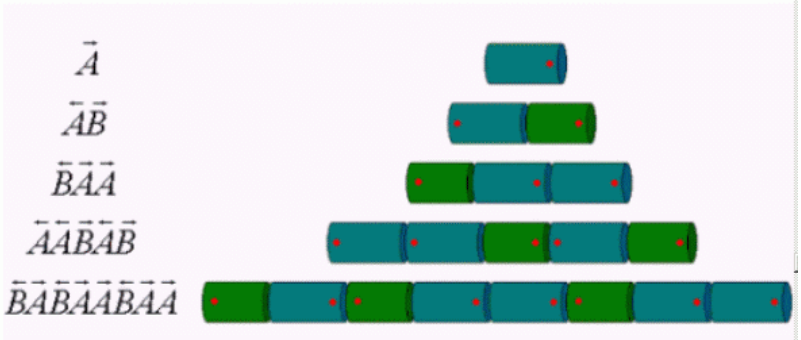
ω : $\vec{A}$

p_1 : $\vec{A} \rightarrow \vec{A}\vec{B}$

p_2 : $\vec{A} \rightarrow \vec{B}\vec{A}$

p_3 : $\vec{B} \rightarrow \vec{A}$

p_4 : $\vec{B} \rightarrow \vec{A}$



Simultane Ersetzung („parallel rewriting")

Fortgeschrittene Softwaresysteme zur Umsetzung:

- cpfg / LStudio
  http://www.cpsc.ucalgary.ca/projects/bmv/index.html
- Grogra http://www.grogra.de
- Graphtal
- LParser http://home.wanadoo.nl/laurens.lapre/lparser.htm

L-Systeme arbeiten *stringbasiert*.
Erweiterungen: Wörter aus parametrisierten Zeichen (Modulen); Graph-Grammatiken; *map*-L-Systeme und *cellwork*-L-Systeme.

Grundversion gut für alle Strukturen mit *lokal 1-dimensionalem Grundgerüst* (Verzweigungssysteme).

Definition:
Ein (kontextfreies, nichtparametrisches) *L-System* ist ein Tripel (Σ, α, R), darin ist

- Σ eine nichtleere Menge von Zeichen (das *Alphabet*),
- α ein Element von Σ^*, das *Startwort* oder *Axiom*,
- R eine nichtleere Teilmenge von $\Sigma \times \Sigma^*$, die Menge der *Produktionsregeln* (generative Regeln).

Ein *Ableitungsschritt* eines Wortes $\beta \in \Sigma^*$ besteht aus der Ersetzung aller Zeichen in β, die in linken Regelseiten von *R* vorkommen, durch die entsprechenden rechten Regelseiten. Man vereinbart: Zeichen, auf die keine Regeln anwenbar sind, werden unverändert übernommen.

Ergebnis zunächst nur:
Ableitungskette von Wörtern, die sich durch iterierte Anwendung des *rewriting*-Vorgangs aus dem Startwort ergeben.

$$\alpha \rightarrow \sigma_1 \rightarrow \sigma_2 \rightarrow \sigma_3 \rightarrow ....$$

In der theoretischen Informatik betrachtet man die Sprachen, die von den so erhältlichen σ_i gebildet werden, ihre Abschlusseigenschaften, Relationen zur Chomsky-Hierarchie...

was für die Morphologie-Modellierung noch fehlt:
eine Semantik (= *geometrische Interpretation*)

füge zu obiger Def. hinzu:
eine Abbildung, die jedem Wort aus Σ^* eine Teilmenge des R^3 zuordnet

dann: "interpretierte" L-System-Abarbeitung

$$\begin{array}{ccccccc} \alpha & \rightarrow & \sigma_1 & \rightarrow & \sigma_2 & \rightarrow & \sigma_3 \rightarrow .... \\ & & \downarrow & & \downarrow & & \downarrow \\ & & S_1 & & S_2 & & S_3 \quad .... \end{array}$$

S_1, S_2, S_3, ... können als Generationen oder als Entwicklungsstufen eines belebten Objekts (Pflanze, Biotop...) interpretiert werden.

Als Interpretationsabbildung wird meistens gewählt:

Turtle geometry ("Schildkrötengeometrie")
befehlsgesteuertes, lokales Navigieren im 2D- oder 3D-Raum

- Abelson & diSessa 1982
- vgl. Sprache "LOGO"

Verwandtschaft des Ansatzes zu "Kettencode-Bildsprachen"
(dort aber meist Befehle mit globaler Bedeutung: "gehe nach unten", "gehe nach rechts" etc.)

in turtle geometry in "Reinform" nur lokale Information und Orientierung
aber: "verwässert" durch Zusatzbefehle, z.B. für Tropismen (Geotropismus, Heliotropismus...): Ausrichtung der Orientierung an festen Richtungen oder Objekten

"Turtle": Zeichen- oder Konstruktionsgerät (virtuell)

- speichert (grafische und nicht-grafische) Informationen
- mit Stack assoziiert
- aktueller Zustand enthält z.B. Information über aktuelle Liniendicke, Schrittweite, Farbe, weitere Eigenschaften des als nächstes zu konstruierenden Objekts

Befehle (Auswahl):

F "Forward", mit Konstruktion eines Elements (Linienstück, Segment, Internodium einer Pflanze...) benutzt wird die aktuelle Schrittweite für die Länge

f forward ohne Konstruktion (move-Befehl)

L(x) ändere die aktuelle Schrittweite (Länge) zu *x*

L+(x) inkrementiere die aktuelle Schrittweite um *x*

L*(x) multipliziere die aktuelle Schrittweite mit *x*

D(x), D+(x), D*(x) analog für die aktuelle Dicke

RU(45) Drehung der *turtle* um die "up"-Achse um 45°

RL(...), RH(...) analog um "left" und "head"-Achse

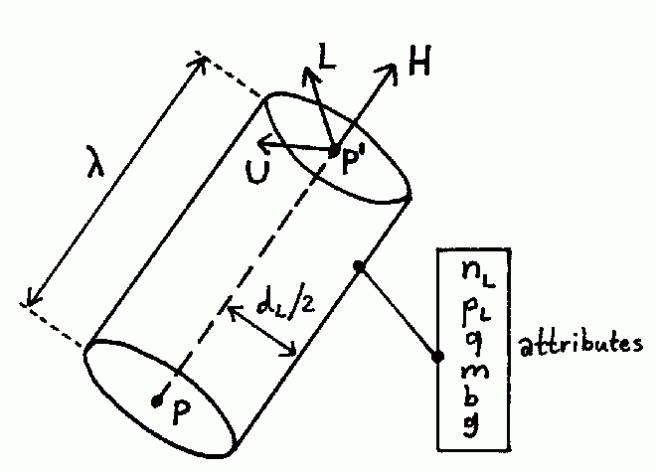
up-, *left*- und *head*-Achse bilden ein orthonormales Rechtssystem, das von der turtle mitgeführt wird

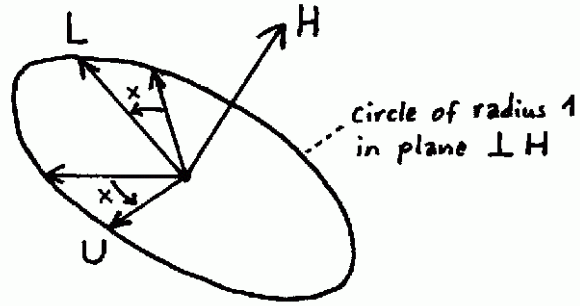
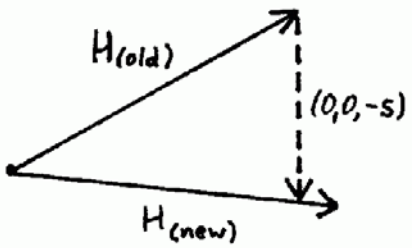
RV(x) Rotation "nach unten" mit durch *x* vorgegebener Stärke

+, – Abkürzung für **RU**(φ) und **RU**(–φ) mit einem festen Winkel φ

(Prusinkiewicz und Lindenmayer verwenden noch weitere Kurzformen für Befehle)

Wirkung des **F**-Befehls (aktuelle Schrittweite ist λ):



Wirkung von **RH**:	Wirkung von **RV** auf die *head*-(Vorwärts-) Richtung:
	

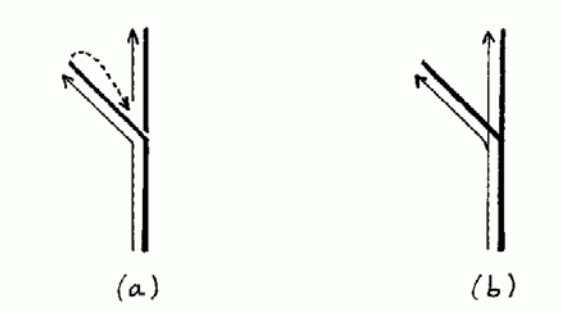
Strings aus diesen Symbolen werden sequenziell abgearbeitet.

Verzweigungen: Realisierung mit Stack-Befehlen

[lege aktuellen Zustand auf Stack

] nimm Zustand vom Stack und mache diesen zum aktuellen Zustand (Ende der Verzweigung)

Interpretation der Klammern sequenziell und parallel möglich (Turtle steht z.B. für pflanzliches, teilungsaktives Gewebe = Meristem)



Der Turtle-Befehlsvorrat wird zu einer Untermenge der Symbolmenge Σ des L-Systems.

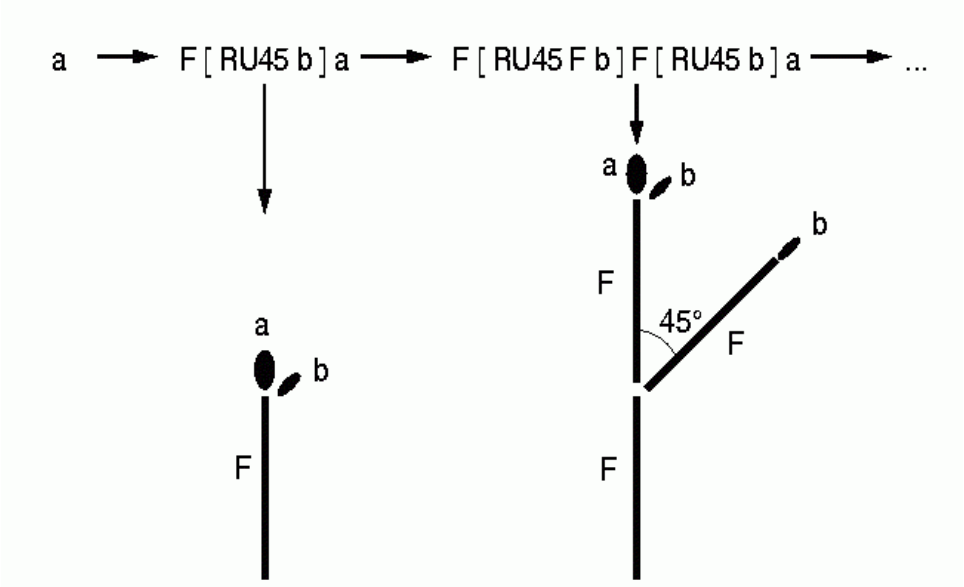
Zuerst Abarbeitung des L-Systems (String-Erzeugung), dann Interpretation der erzeugten Wörter durch die Turtle. Symbole, die nicht Turtle-Befehle sind, werden von der Turtle ignoriert.

Beispiel:

Regeln

a → F [RU45 b] a,
b → F b

Startwort **a**



(**a** und **b** werden normalerweise nicht geometrisch interpretiert.)

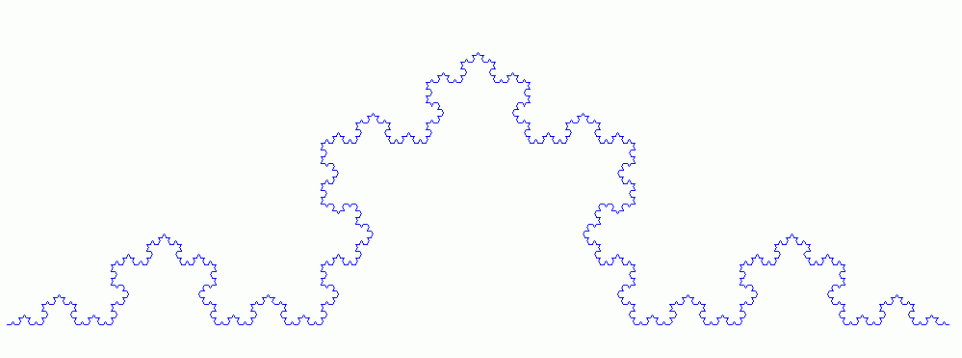
Weitere Beispiele:

Koch-Kurve:

```
\angle 60,
* → RU90 a F,
a → a L*0.3333,  /* Skalierung */
F → F - F + + F - F
```

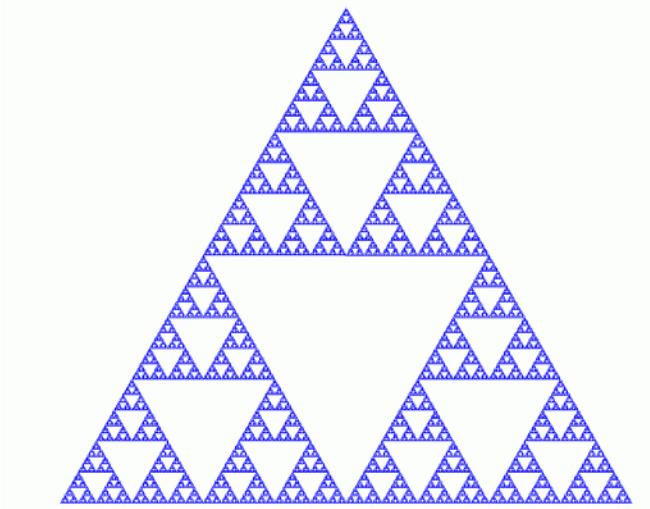
jedes Linienstück wird durch 4 neue Linienstücke ersetzt (3. Regel); Skalierung durch Hilfssymbol, welches sich in jedem Schritt reproduziert (2. Regel).
Das Startwort ist hier " ***** ".
"**\angle**" spezifiziert den Winkel für "**+**" und "**–**".

Ausgabe nach 6 Schritten:



Sierpinski-Dreieck (Realisierung als geschlossene Kurve, Verwendung von Hilfssymbol **x** für Insertion des inneren Dreiecks):

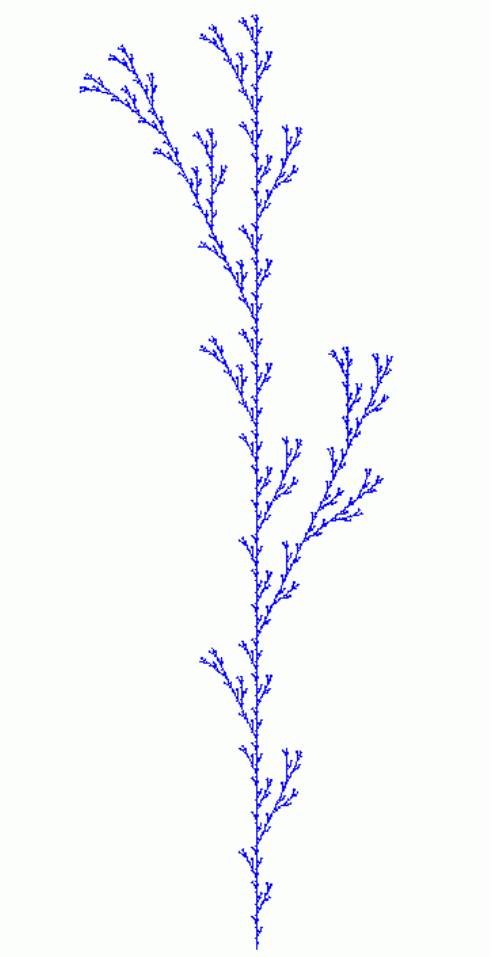
```
\angle 60,
* → RU90 b F x F - - F F - - F F,
F → F F,
x → - - F x F + + F x F + + F x F - -,
b → b L*0.5
```



Verzweigung, "Pseudo-Pflanze":

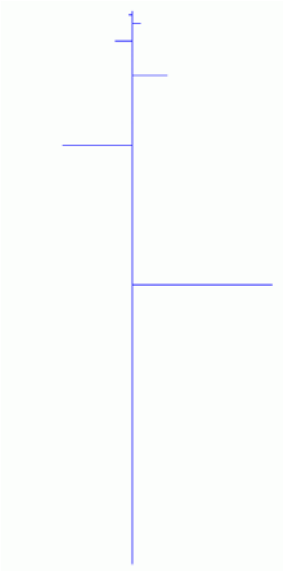
\angle 25.7,
F → F [+ F] F [– F] F

Ergebnis nach 7 Schritten:

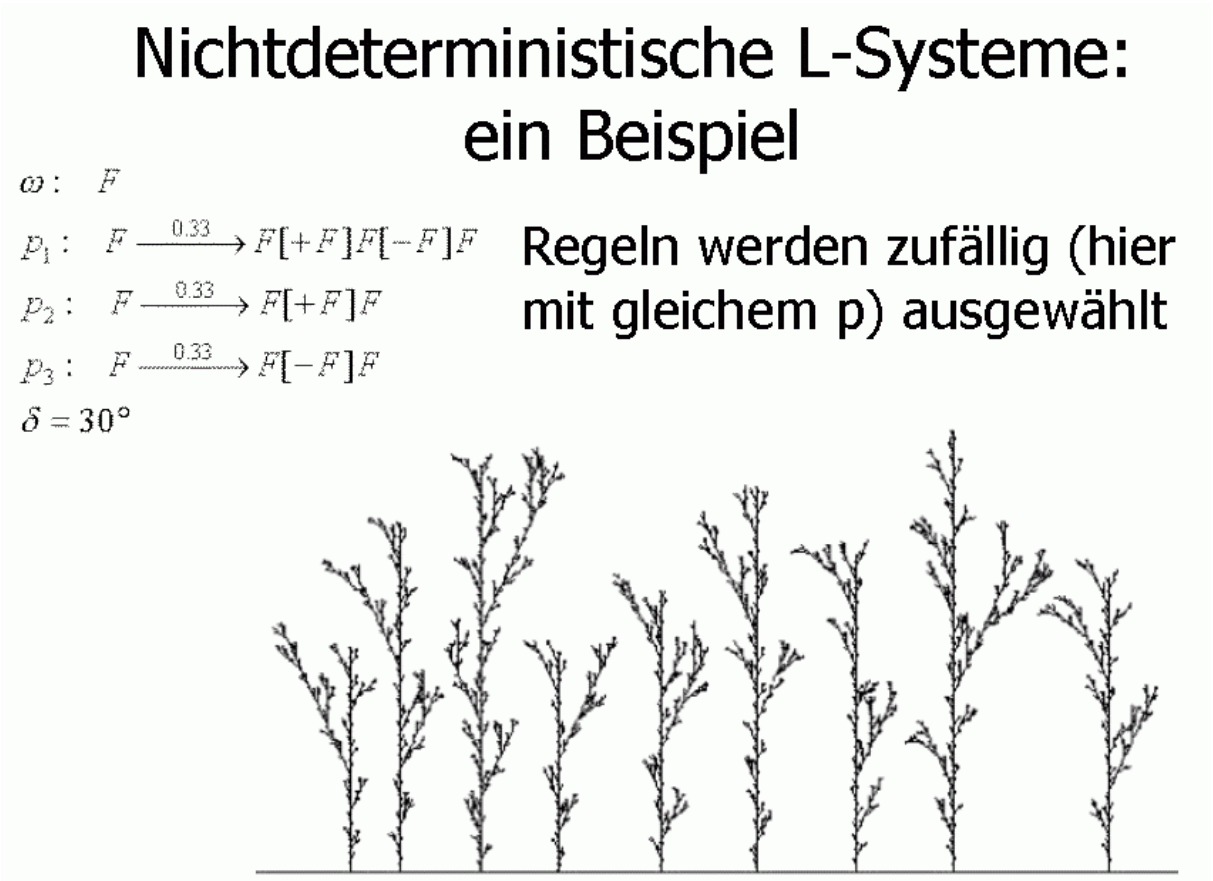


Verzweigung, alternierende Zweigstellung und Verkürzung:

*** → F a,**
a → L*0.5 [RU90 F] F RH180 a

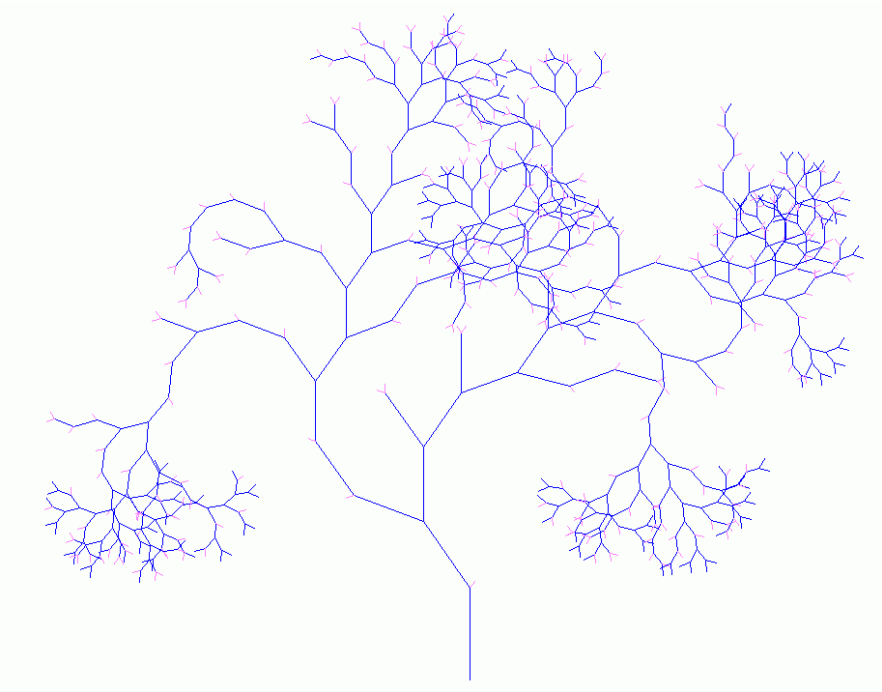


Stochastische L-Systeme: Einbau von Zufallsauswahl



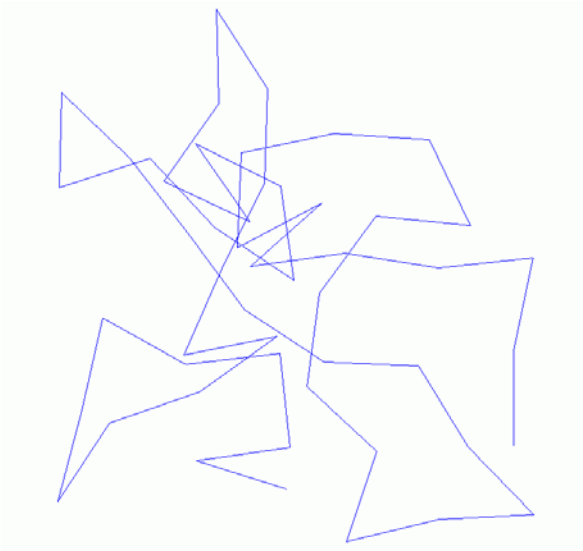
Verzweigung, Absterbemöglichkeit (stochastisches L-System):

```
* → F L*0.9 [ RU35 * ] RU-35 * ?0.65,
* → P2 L10 F ?0.35
```



Irrflug (Brownsche Bewegung in 2D), Verwendung einer gleichverteilten Zufallsvariablen:

```
\var x uniform 0 360,
* → F a,
a → RU(x) F a
```



Erweiterung des Konzepts:
Lasse reellwertige Parameter nicht nur bei Turtle-Kommandos wie "**RU45**" zu, sondern bei allen Symbolen

→ *parametrische L-Systeme*, operieren auf "Moduln" statt auf Wörtern
beliebig lange, endliche Parameterlisten
Parameter werden bei Regel-Matching mit Werten belegt

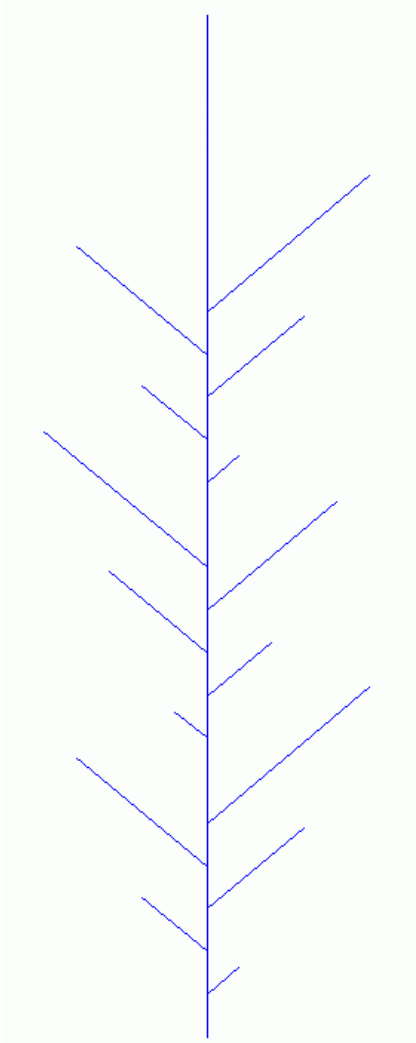
Beispiel:
Regel **`a(x, y)`** → **`F(x^3+10) b(2*y)`**
vorliegendes Wort "**`a(2, 3)`**"
nach einer Regelanwendung: **`F(18) b(6)`**

Parameter können in Konditionen abgeprüft werden
(Konditionen mit C-Syntax):
`(x >= 17 && y != 0) a(x, y)` → ....

Beispiele:

Verwendung des Wiederholungsoperators "&" und von parametrisierten Symbolen
Länge des abzweigenden Astes abhängig von Position
(Bauprinzip bei vielen Gehölzen!) – "Akrotonie":

```
\var i index,
* → a(5),
a(n)→ &(n) < F [ RU50 lat(i) ] RH180 > F a(n),
lat(j) → L+(j*100) F
```



Beispiel für ein L-System mit mehreren Hilfssymbolen, die unterschiedliche Meristem-Typen einer Pflanze repräsentieren:

```
\var x0 uniform 0 360,
\var x1 normal 0 15,
\var x2 uniform -10 0,
\var x3 uniform -25 25,

* # [ P14 t0 ] P4 &6 < [ RH(x0) P5 ul' ] > L*0.6 u0,
t0 # xt F D RH137.5 [ RL80 L*0.5 P2 k(1) s1 ]
     [ RH180 RL80 L*0.5 P2 k(1) s1 ] t0,
s1 # xs F D [ RH25 RU60 $ L*0.7 s2 ] [ RH-25 RU-60 $ L*0.7 s2 ] s1,
s2 # xs F D,
xt # xt D+3,
xs # xs D+2,
(t < 6) k(t) # k(t+1),
(t = 6) k(t) # %,
u0 # RG RH(x0) RU(x1) xt F D
     [ L*1.1 k(1) P15 u1 ] [ L*1.1 k(1) P15 u1' ] u0,
ul' # ul,
```

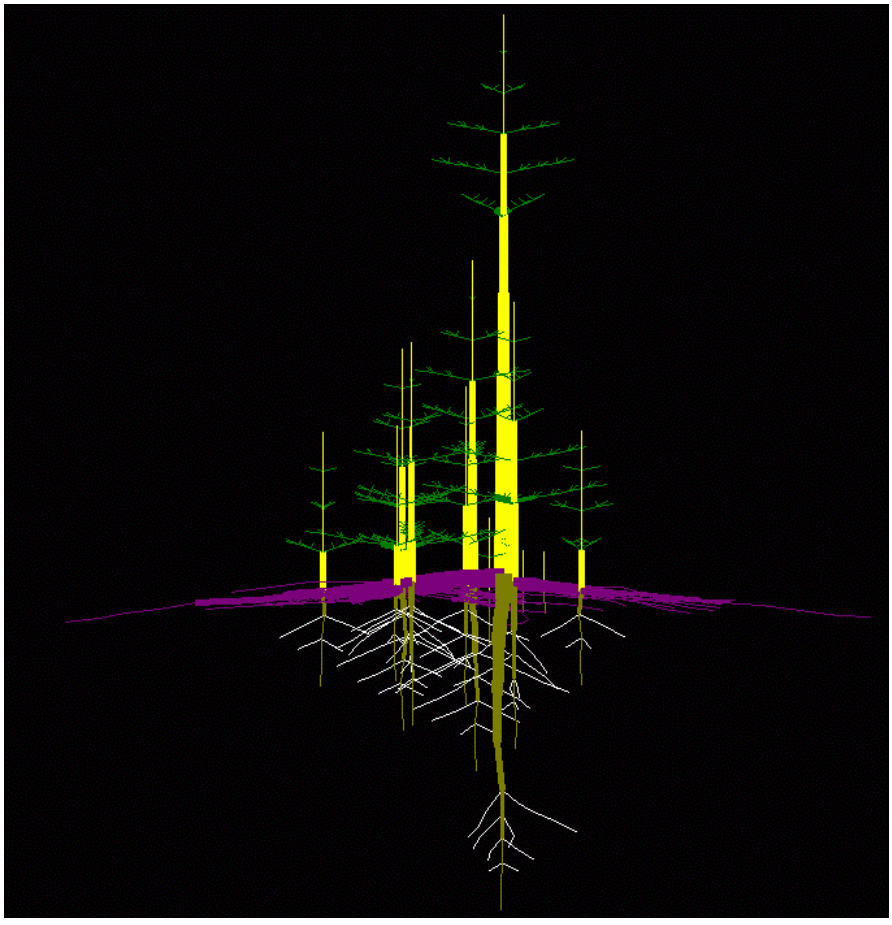
```
ul # RG RL90 RL(x2) RU(x3) xs F D a1 ul,
u1 # u2,
u1' # u2',
u2 # RG RL70 RL(x2) RU(x3) xs F D u2,
u2' # RG RH180 RL70 RL(x2) RU(x3) xs F D u2,
a1 # a2,
a2 # a3,
a3 # [ RG RU180 * ] ?0.2,
a3 # z ?0.8
```

Beachte:

- **k(t)** als "Uhrensymbol" (**t** wird in jedem Schritt um 1 inkrementiert)
- Verwendung eines "Cut-Operators" **%**
- Wiederauftauchen des Startsymbols "*****" in der vorletzten Regel: "Reiteration", d.h. Reproduktion des gesamten Bauplans der Pflanze (in diesem Fall aus Wurzel-Meristemen, "Wurzelbrut")

Austriebs- und Absterbewahrscheinlichkeiten können in der Praxis aus botanischen Messungen entnommen werden

Ergebnis nach 15 Schritten:

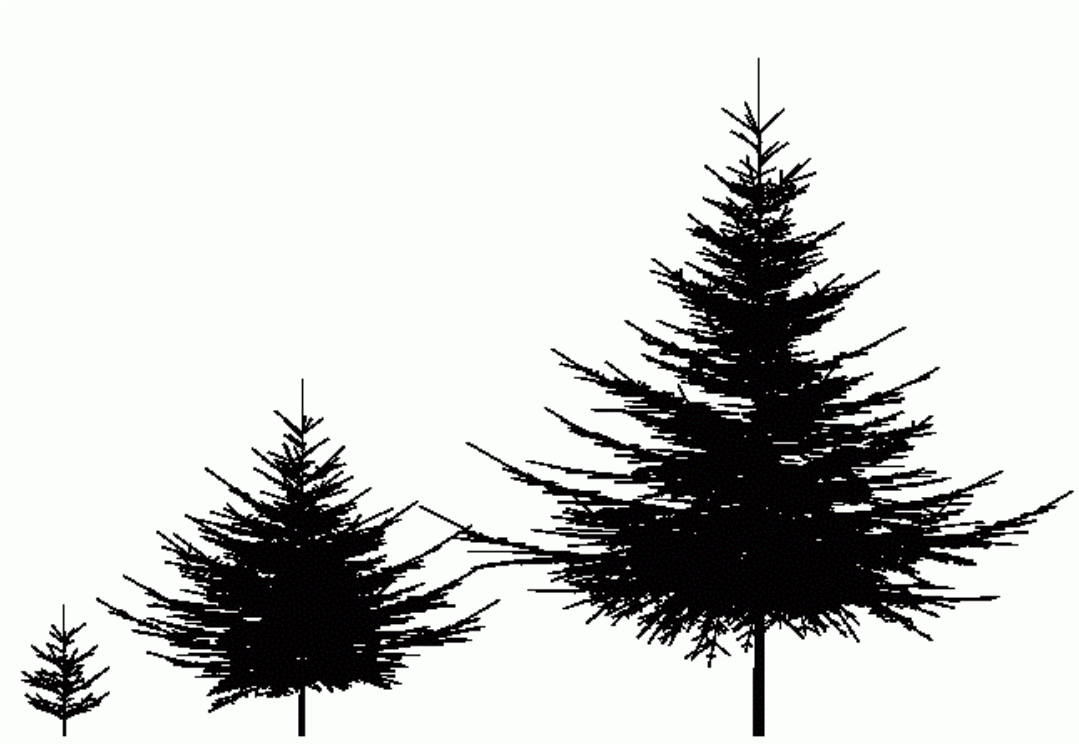


zur Modellierung konkreter botanischer Objekte:

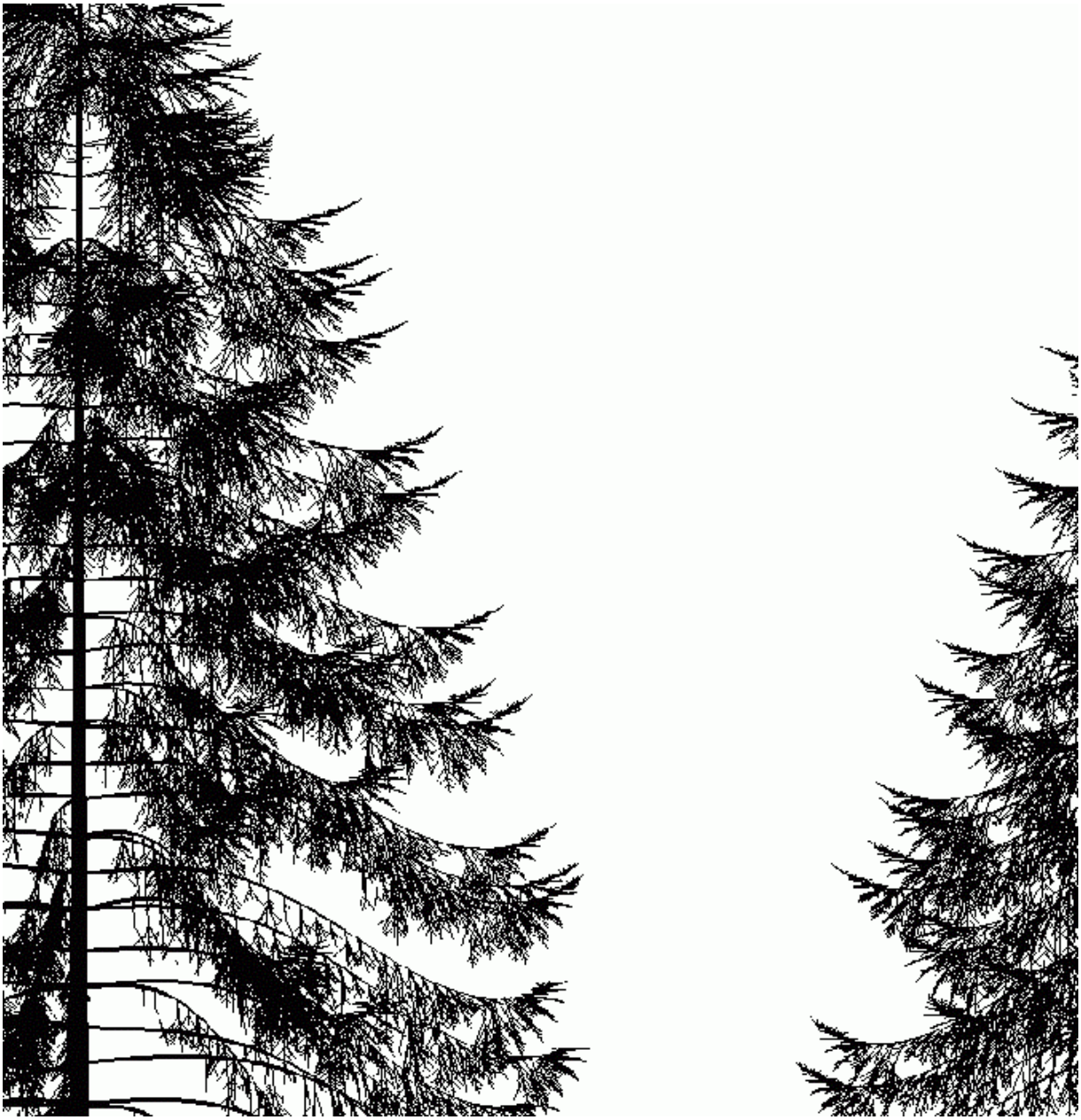
Vorgehen bei L-Systemen

- Analyse des Objekts in mehreren Zuständen
  in der Natur (Beobachtung) und / oder im Labor
- Auf **nichtformalem** Weg die Regeln aufstellen
- Regeln und Anfangszustand in ein L-System verwandeln
- Simulation laufen lassen, die eine lange Zeichenkette erzeugt
- Resultat in eine graphische Ausgabe übersetzen
- Das Bild (oder mehrere Bilder von verschiedenen Stadien)
  mit dem Verhalten des wirklichen Objekts vergleichen
- Evtl. Korrekturen vornehmen und
  die erforderlichen Schritte wiederholen.

Beispiel Fichte:

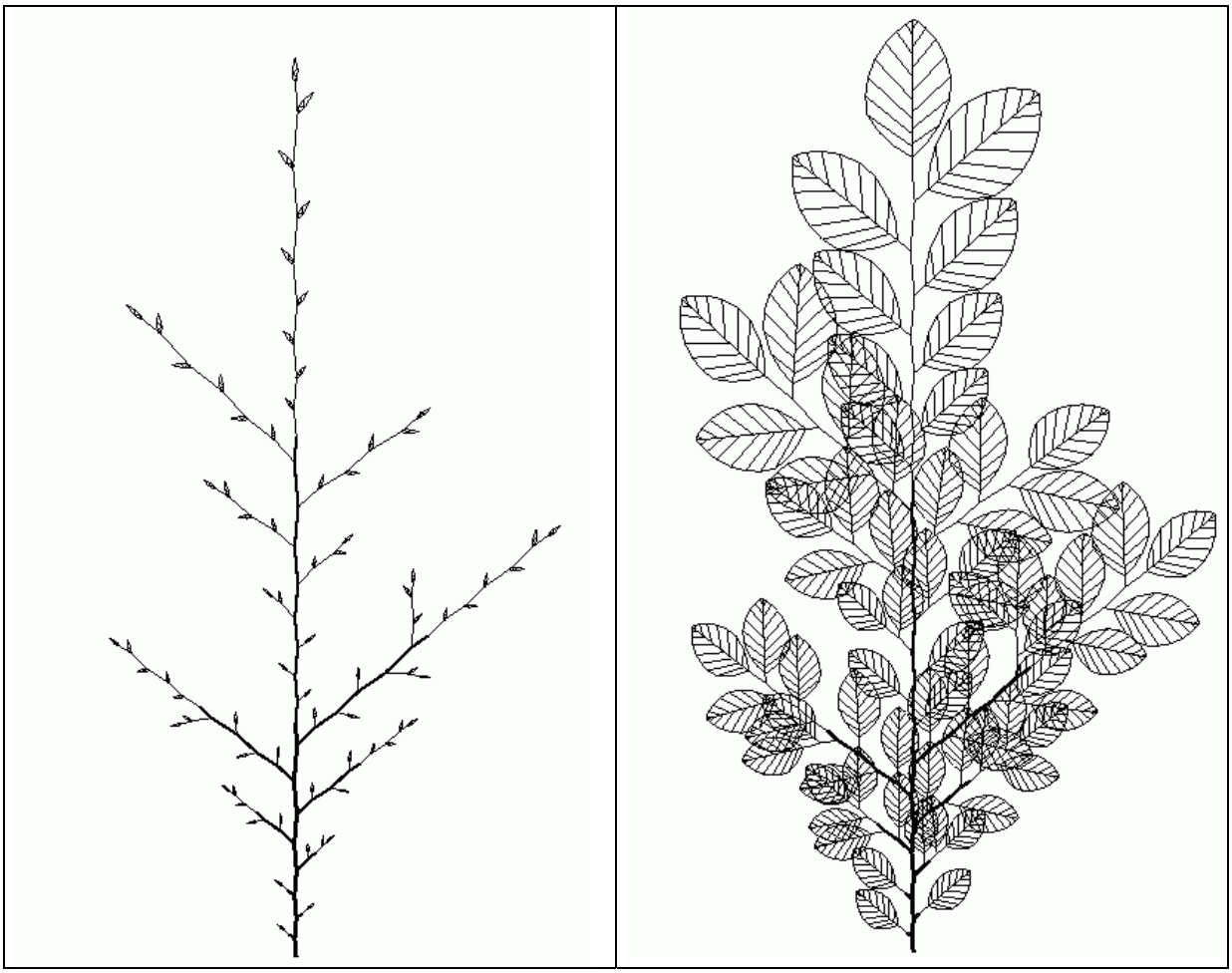


(basierend auf Messungen am realen Objekt)



Beispiel Minze (von Prusinkiewicz & Lindenmayer):

Beispiel Buchenzweige:



Nützliche Erweiterung des Formalismus:
Einführung einer zusätzlichen Regelmenge
"Interpretationsregeln"
wirken nicht auf den String der nächsten Generation
Verwendung zum Zeichnen: Vorstufe der Turtle-Interpretation
z.B. für die Knospen und Blätter beim obigen Buchenzweig

$$\begin{array}{ccccccc} \alpha \rightarrow & \sigma_1 & \rightarrow & \sigma_2 & \rightarrow & \sigma_3 & \rightarrow .... \\ & \Downarrow & & \Downarrow & & \Downarrow & \\ & \sigma_1' & & \sigma_2' & & \sigma_3' & .... \\ & \downarrow & & \downarrow & & \downarrow & \\ & S_1 & & S_2 & & S_3 & .... \end{array}$$

Doppelpfeile ($\Downarrow$) bedeuten Anwendung der Interpretationsregeln

Nachteil der bisher vorgestellten L-Systeme:
Kontrolle nur durch Vorgänger-Symbol ("lineage control") oder stochastisch

⇒ fehlende Interaktion innerhalb des modellierten Objekts oder mit der Umwelt

⇒ Determinismus oder stochastische Modelle ohne kausale Komponenten

Abhilfe: Einführung von *Sensitivität* bei der Regelanwendung.

(a) Kontextsensitivität

(schon altes Konzept, Beispiele bereits bei Lindenmayer...):

Abhängigkeit einer Regelanwendung vom linken und / oder rechten Kontext im String:
leftcontext **< a >** *rightcontext* $\rightarrow \beta$.
stringbasiert!

Verwendung:

- Weiterleitung von Signalen innerhalb der modellierten Struktur

  s < a > → a s,

  s → , /* leeres Wort */
- Konzentration von Substanzen (Hormonen)
- Bewegung von Objekten (z.B. Insekten auf der Pflanze)

Beispiel: Entwicklung von Blütenständen häufig hormonal gesteuert

häufig: Zusammenwirken von 2 Signalen (von unten und von oben)

- das Modell kann Aufschluss geben, ob für eine in der Natur beobachtete morphologische Sequenz der Blütenbildung 1, 2 oder mehr Pflanzenhormone notwendig sind.

weiteres Beispiel von Prusinkiewicz & Lindenmayer (1990): *Lychnis coronaria*

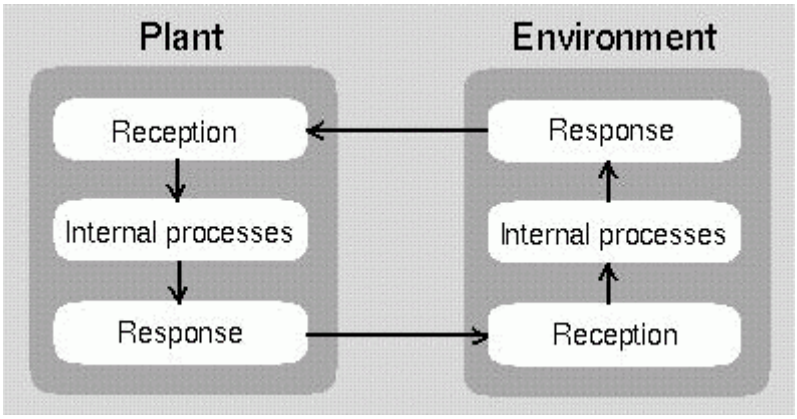
(b) globale Sensitivität
(auch: *environmentally sensitive L-systems*)

- Kommunikation mit der Umgebung über spezielle Kommunikationsmodule oder über sensitive Funktionen
- Regelanwendung hängt (potentiell) von der gesamten, aktuell vorhandenen Struktur im Objektraum (und von eventuellen externen Eingriffen) ab (nicht nur von der String-repräsentation)

- Schnittstelle zu physikalisch oder biologisch basierten Simulationsmodellen (man spricht dann von *open L-systems*)

2 Herangehensweisen:

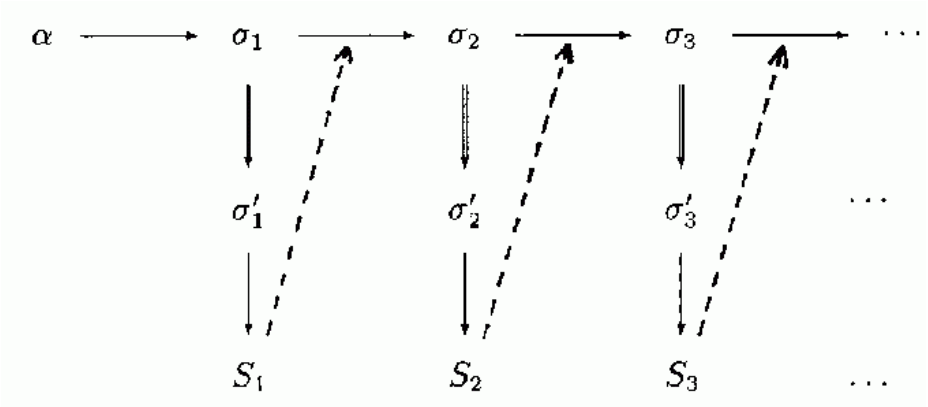
(a) scharfe Trennung von Organismus und Umgebung; der lebende Organismus wird mittels L-System modelliert, die Umgebung mittels anderer Modelle, Implementation als parallele, kommunizierende Prozesse



(Prusinkiewicz)

(b) durch Erweiterung des L-System-Formalismus Aspekte der Umgebung ins L-System "hineinholen"

Prinzip der Anwendung global sensitiver Funktionen:



gestrichelte Pfeile: Informationsfluss von der erzeugten Objektstruktur zur Regelanwendung (Regelauswahl, Parametrisierung) auf die Strings.

⇒ konzeptionell Annäherung an Graph-Grammatiken (dort würden nur noch S_1, S_2 usw. existieren, keine Strings mehr).

Beachte:

- kontextsensitive L-Systeme entsprechen den "selbst-regulierten Mustern" mit endogener Kontrolle,
- global sensitive Systeme den "sehenden Mustern" mit exogener Kontrolle (in der Systematik der Morphogenese-Modelle nach Adrian D. Bell).

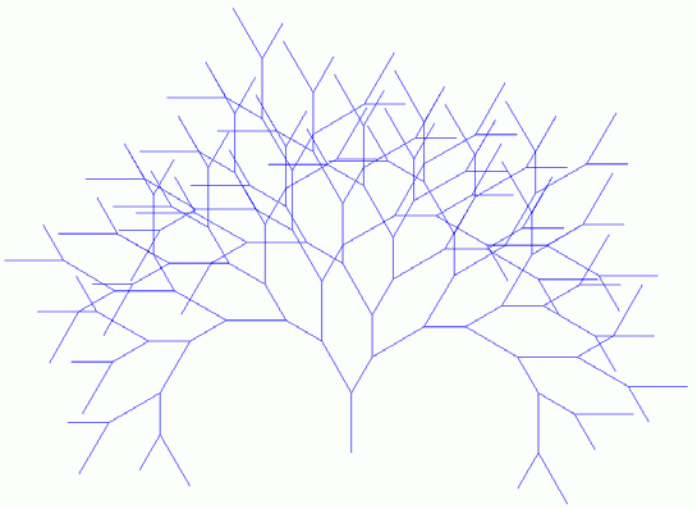
einfaches Beispiel:

Nichtsensitives und global sensitives L-System im Vergleich

nichtsensitive, dichotome Verzweigung:

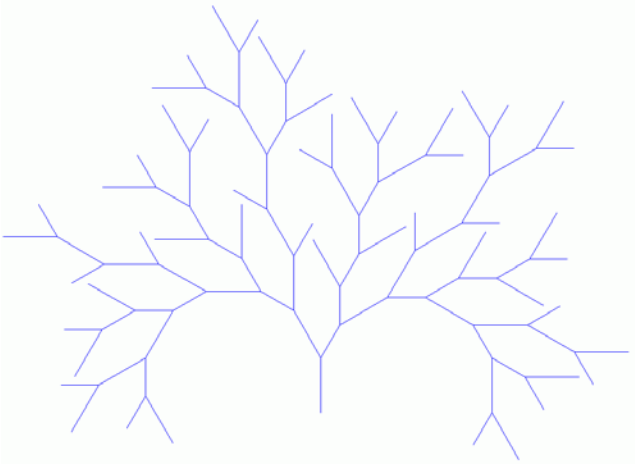
```
\axiom a 1-8,
\angle 30,
a → RH180 F100 [ - b ] + a,
b → RH180 F70 [ - b ] + a
```

Ergebnis:



sensitive Verzweigung mit Abhängigkeit vom Abstand zum nächsten Nachbar-Element (im Objektraum!):

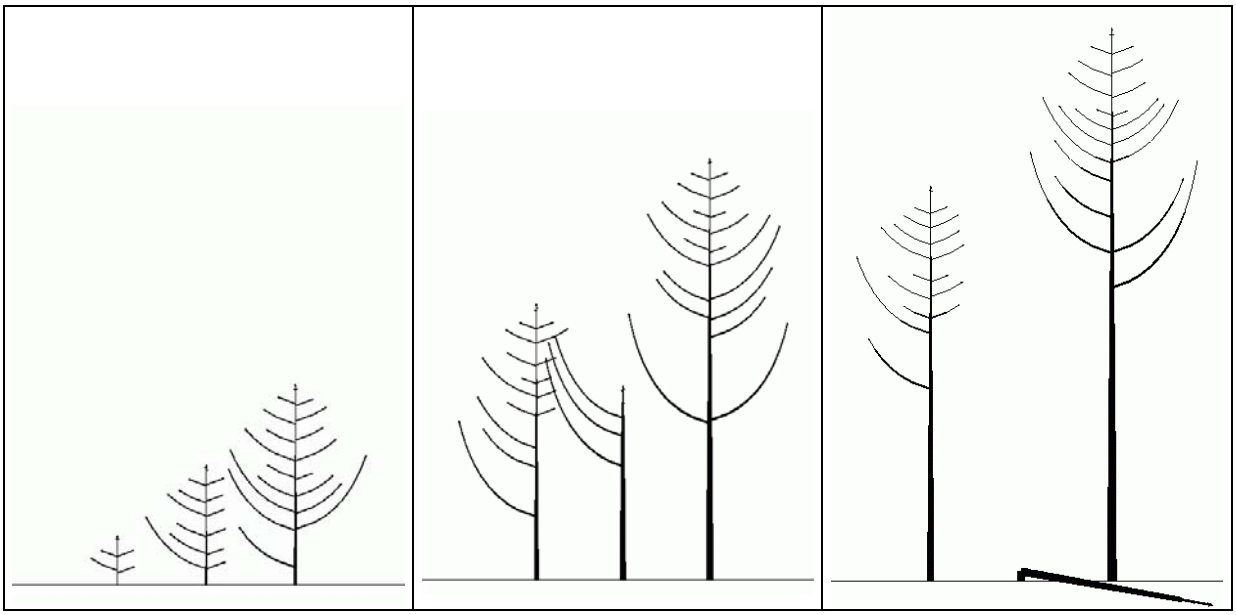
```
\axiom a 1-8,
\angle 30,
\var f function 2 1,
(f(1) > 60) a → RH180 F100 [ - b ] + a,
(f(1) > 60) b → RH180 F70 [ - b ] + a
```



Anwendung in der Pflanzenmodellierung:

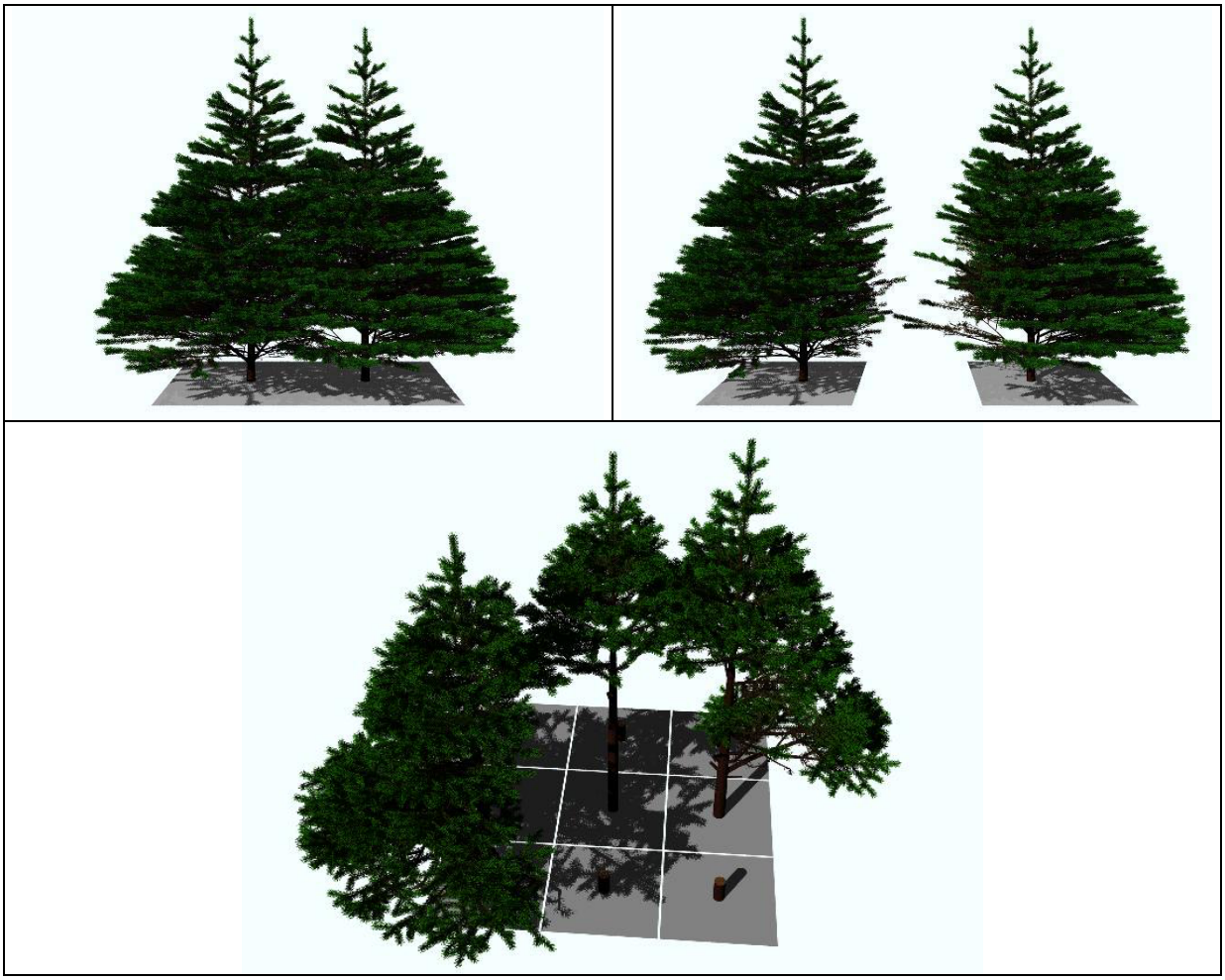
- Dichteabhängigkeit des Wachstums
- Einfluss des Neigungswinkels eines Astes auf den Neuaustrieb
- Einfluss der Beschattung
- Wechselwirkung mit Herbivoren (Tiere)

Beispiel: einfaches Überschattungsmodell (2D)



hier ist das Wachstum in jedem Apex vom "Offen-sein" eines nach oben geöffneten Kegels (bzw. Winkelfeldes) abhängig; zusätzlich werden Äste, die mehrere Schritte nicht gewachsen sind, als "tot" entfernt, und der Baum kippt um, wenn alle seine Äste tot sind.

Wachstum von Bäumen unter Konkurrenzbedingungen, mit *open L-system* realisiert
(P. Prusinkiewicz et al.)



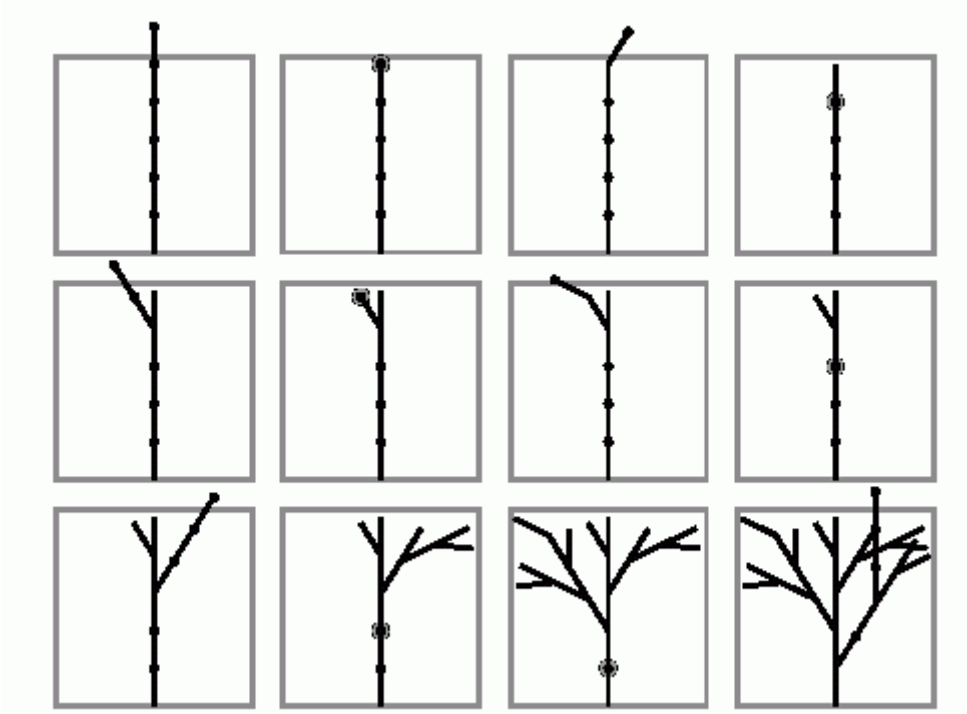
Selbstauslichtung (Astabwurf im Bestandesschatten) und asymmetrische Kronenformen als emergente morphologische Erscheinungen

weitere Beispiele später, bei "Struktur-Funktions-Modellen".

äußere Eingriffe: Beschneiden von Gehölzpflanzen, z.B. im Gartenbau, um eine bestimmte Form zu erzielen

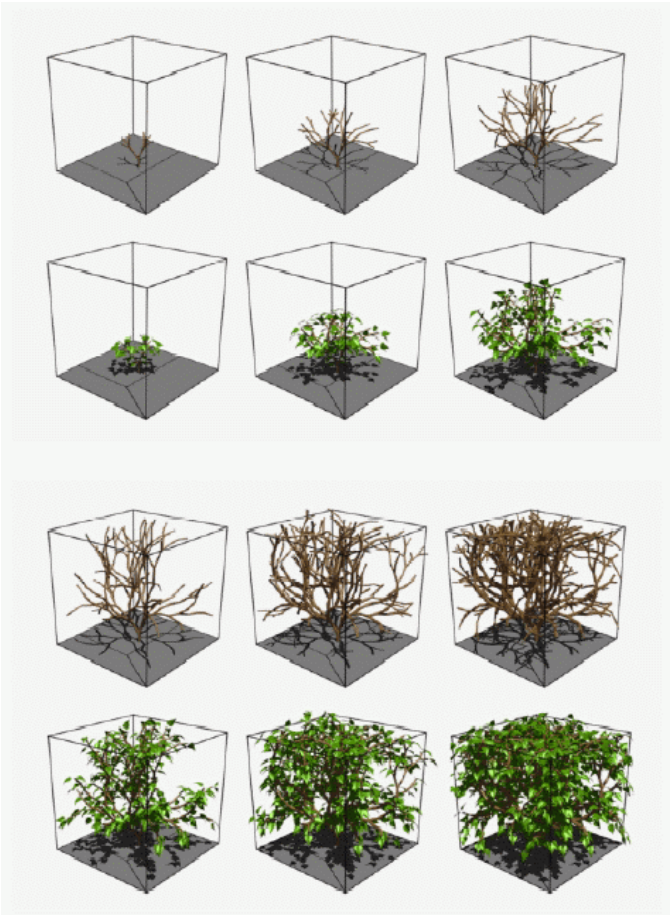
- botanischer Mechanismus: Induktion der Bildung von Ersatztrieben
- Modell: sensitives L-System, das Überschreiten einer räumlich vorgegebenen Grenze registriert und wachstumsanregendes Signal induziert, dieses bringt schlafende Knospen zum Austrieb

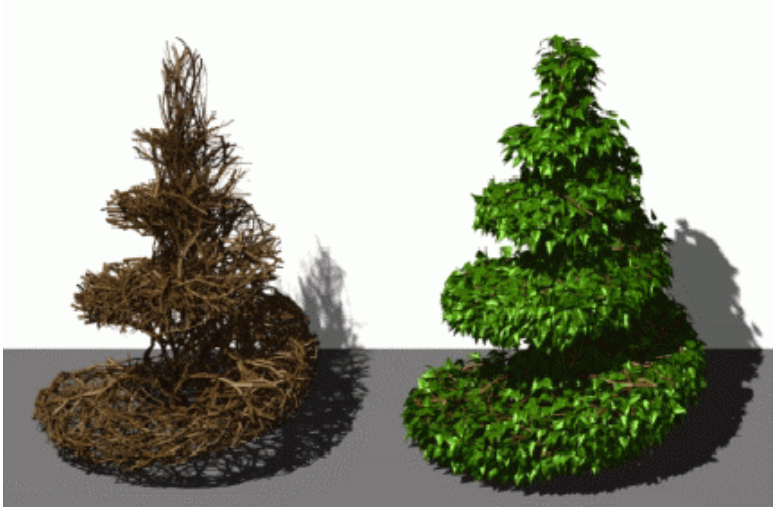
aus Prusinkiewicz et al. (1994):

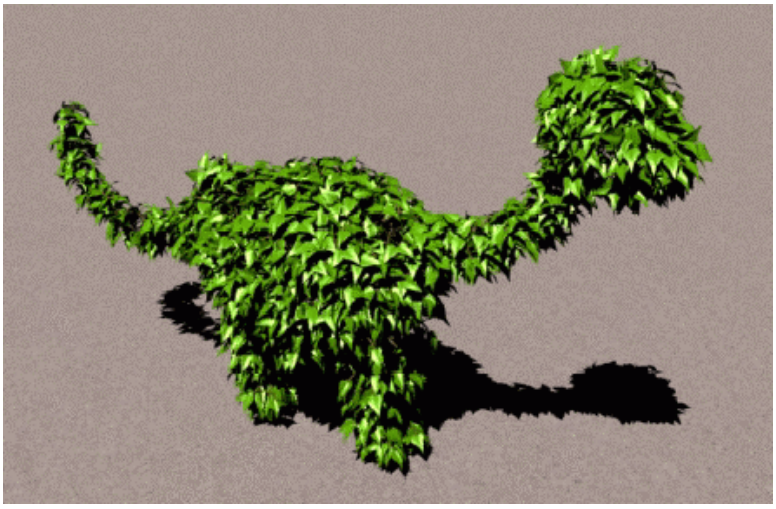


(kleine Kreise: schlafende Knospen, großer Kreis: Wachstumssignal – wandert von oben nach unten bis zur nächsten schlafenden Knospe)

3D-Version, gerendert:



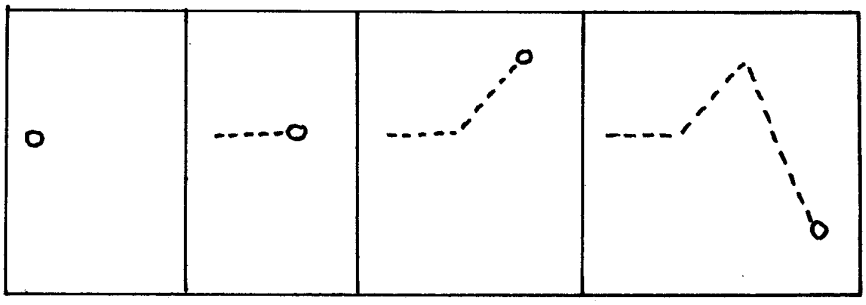




Moseley Old Hall Garden:

Beschränkung der L-System-Modelle auf ortsfeste Organismen: wie kann diese überwunden werden?

Verwendung des **f**-Befehls für Bewegungen:



Nachteil: die Folge von "**f**"s wird im String immer mitgeführt

Abhilfe: Verwendung von "Metaregeln", die in diesem Fall z.B. zwei aufeinanderfolgende, parametrisierte **f**-Kommandos durch ihre Vektorsumme ersetzen
(bisher noch nicht in L-System-Software realisiert!)

weitere sinnvolle Erweiterungen:

- Mengen statt Strings (einfachere, ungeordnete Strukturen, z.B. Substanzen in einer Lösung)
- Regeln für Interaktion (Austausch von Botschaften zwischen bestimmten Symbolen)

Timed L-systems (zeitverstetigte L-Systeme)

Überwindung der Beschränkung auf diskrete Zeit:

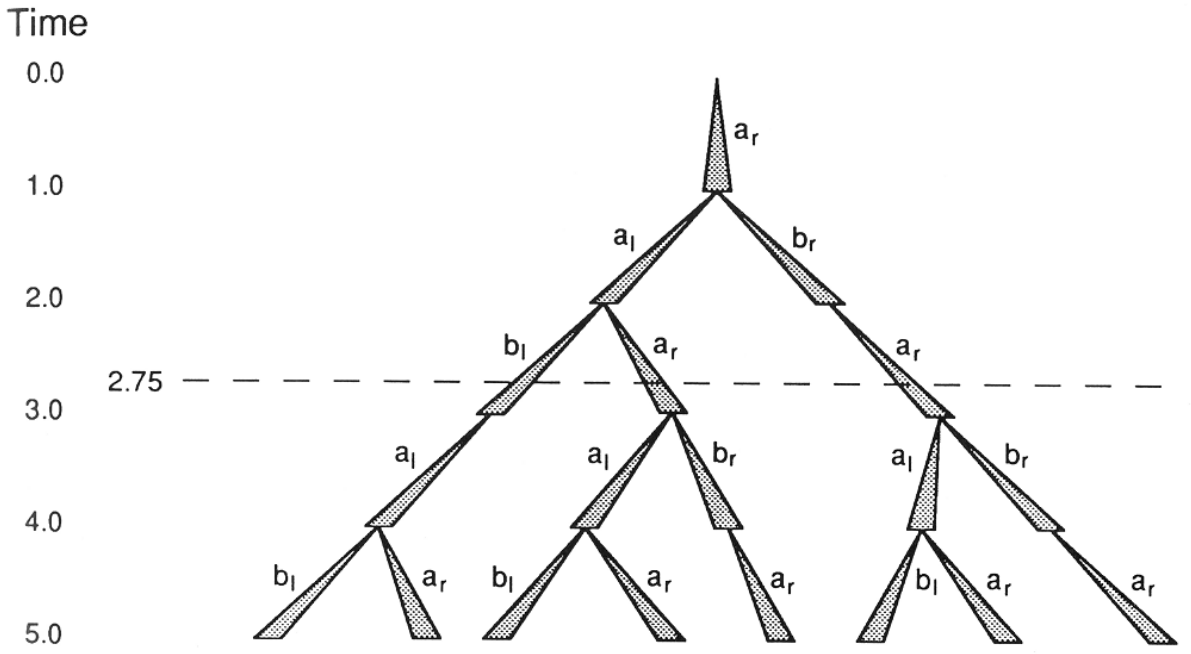
- jedes Symbol erhält als (zusätzlichen) Parameter einen Zeit-Wert
- dieser wird im Verlauf der Simulation stetig "hochgezählt"
- Produktionsregeln werden ausgelöst, wenn das "terminale Alter" eines Symbols erreicht ist

formale Def. siehe Prusinkiewicz & Lindenmayer (1990)

Beispiel:

$$
\begin{array}{lll}
\omega: & (a_r, 0) & \\
p_1: & (a_r, 1) & \rightarrow (a_l, 0)(b_r, 0) \\
p_2: & (a_l, 1) & \rightarrow (b_l, 0)(a_r, 0) \\
p_3: & (b_r, 1) & \rightarrow (a_r, 0) \\
p_4: & (b_l, 1) & \rightarrow (a_l, 0)
\end{array}
$$

(ω = Startsymbol, p_1, ..., p_4 = Produktionsregeln)



Nachteil: Simulation wird aufwändiger, ein Scheduler muss die Synchronisation überwachen

Überwindung der Beschränkung auf lokal 1-dimensionale Topologien (Verzweigungssysteme):

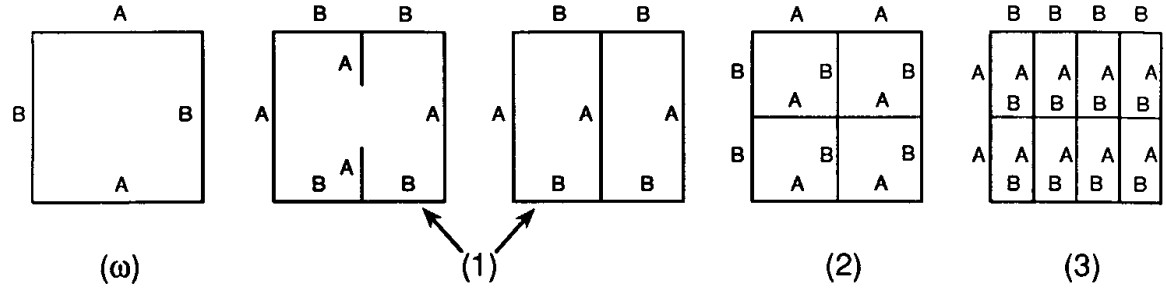
map L-systems, cellwork L-systems

verschiedene Varianten möglich
Beispiel:
Binary propagating map 0L-systems with markers (mBPM0L-systems; Prusinkiewicz & Lindenmayer 1990)

binary: eine Region kann sich in höchstens 2 Tochter-Regionen teilen
propagating: es können keine Kanten gelöscht werden
Marker: legen die Positionen neuer Kanten fest, die eine Region teilen

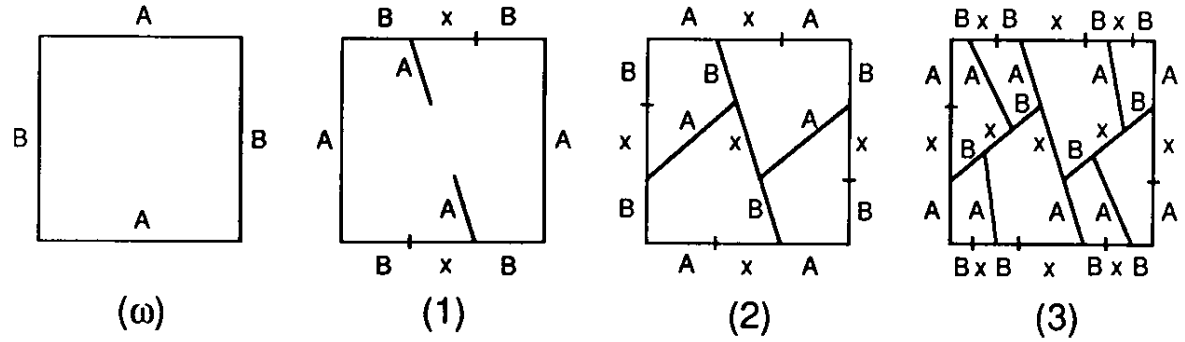
Symbole stehen für markierte, gerichtete Kanten (*edge labels*)
ein Ableitungsschritt besteht aus 2 Phasen: Ersetzung aller Kanten durch Nachfolgekanten und ggf. Marker entsprechend den Produktionsregeln, und Verbinden von Markern, die "matchen", zu neuen Kanten.

Beispiel:
Startwort **ABAB**,
Regeln **A** $\rightarrow$ **B [- A] [+ A] B** und **B** $\rightarrow$ **A**



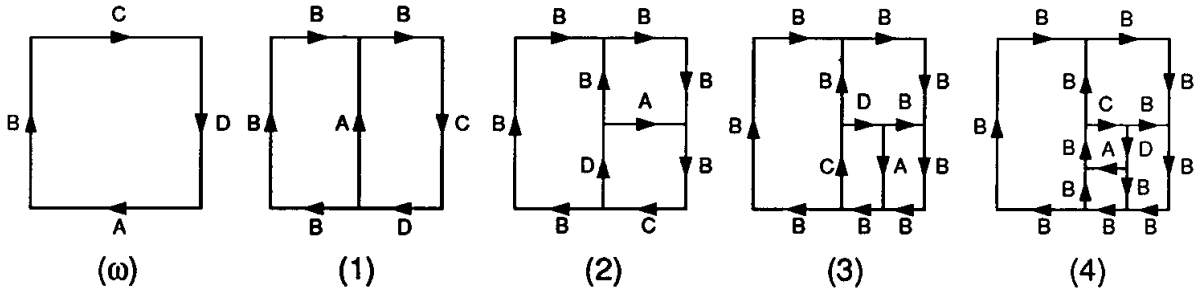
ω : ABAB
p_1 : A $\rightarrow$ B[-A][+A]B
p_2 : B $\rightarrow$ A

2. Beispiel:



ω : ABAB
p_1 : A → B[-A]x[+A]B
p_2 : B → A

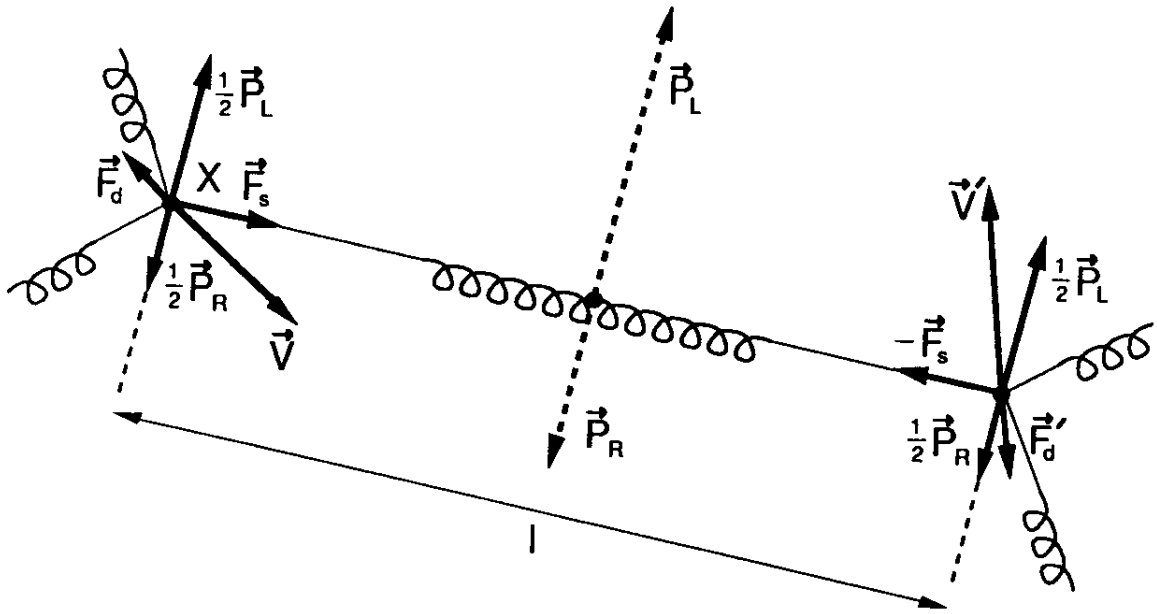
3. Beispiel (mit gerichteten Kanten):



ω : $\vec{A}\vec{B}\vec{C}\vec{D}$
p_1 : $\vec{A}$ → $\vec{D}[-\vec{A}]\vec{B}$
p_2 : $\vec{B}$ → $\vec{B}$
p_3 : $\vec{C}$ → $\vec{B}[-\overleftarrow{A}]\vec{B}$
p_4 : $\vec{D}$ → $\vec{C}$

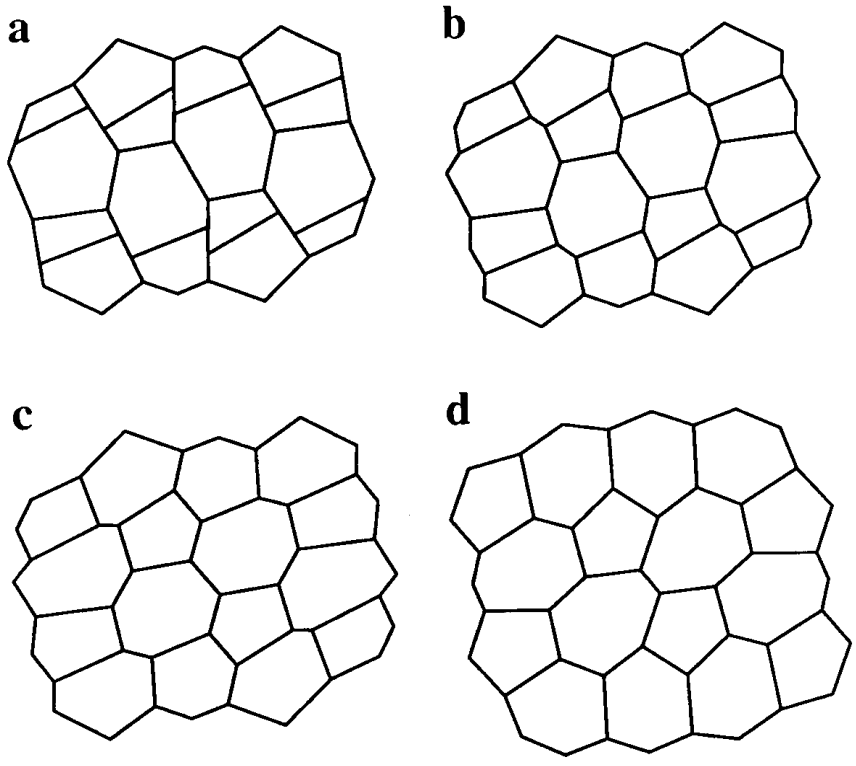
Problem: Geometrie der erzeugten Strukturen ist durch die Regeln nicht festgelegt
(anders als bei gewöhnlichen L-Systemen mit Turtle-Interpretation)

ein möglicher Ansatz: dynamisches Modell, das Kräfte annimmt, die jede Zelle auf ihre Wände (Kanten) ausübt



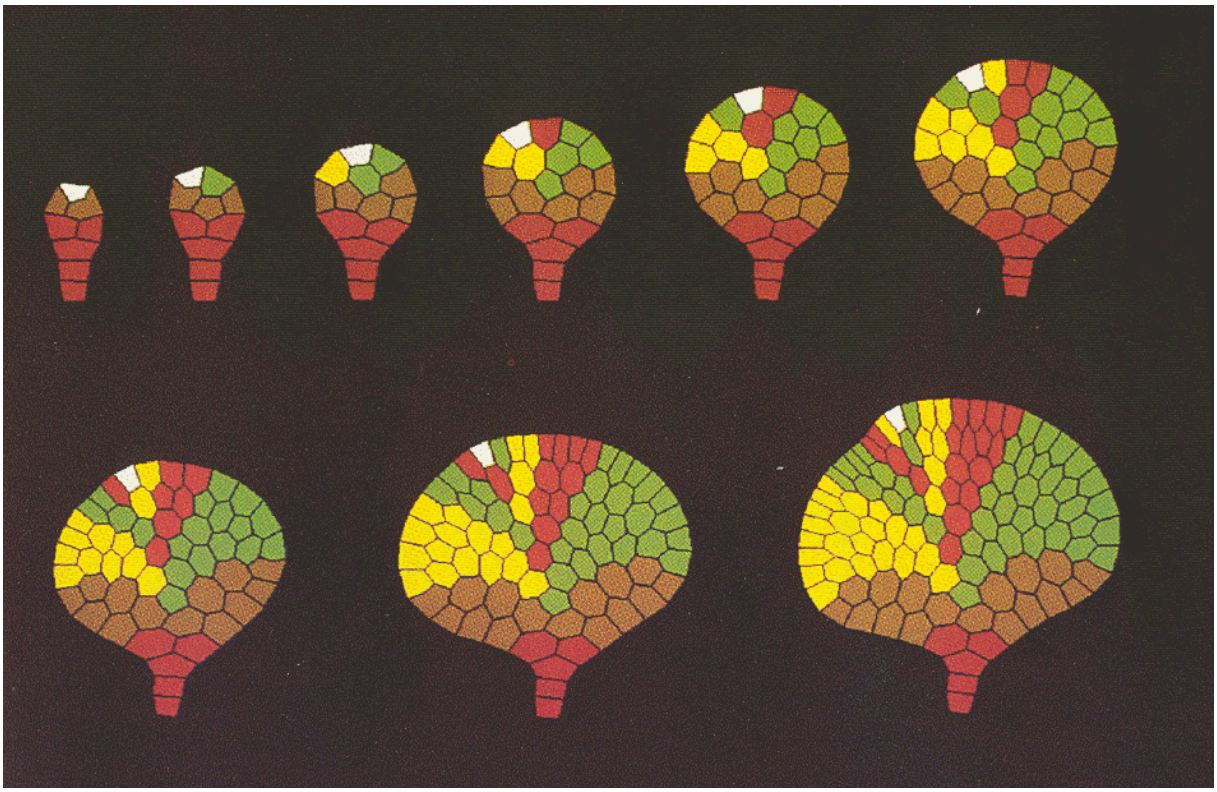
$\Rightarrow$ führt auf System von Differentialgleichungen,
Lösung führt auf Gleichgewichtszustand

Beispiel (Iterationsschritte bei Lösung):

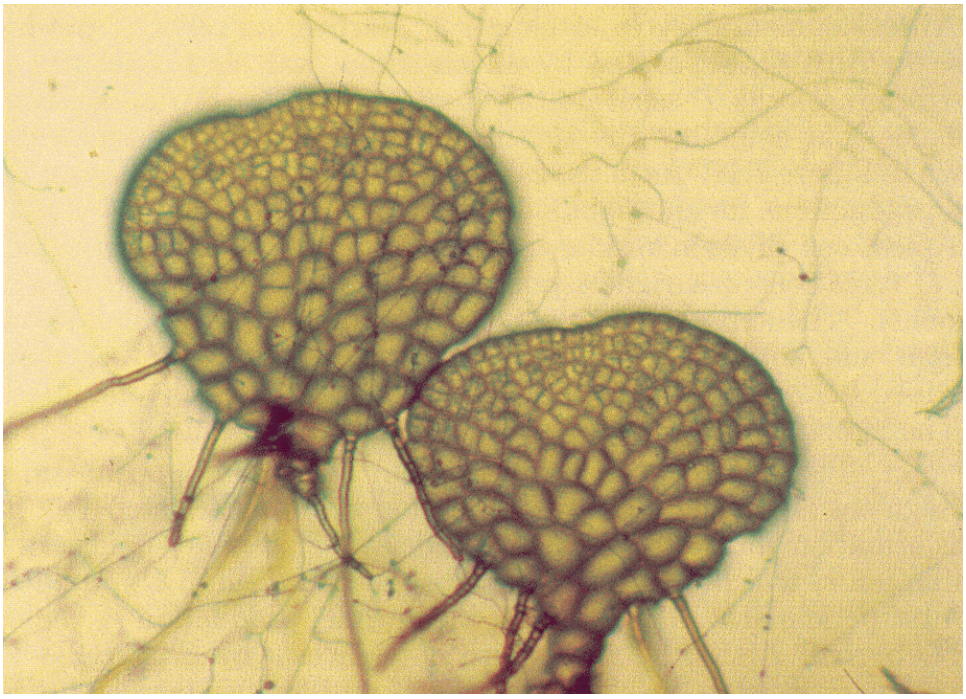


Anwendung: Simulation von Mikroorganismen
hier: *Microsorium linguaeforme* (Farn-Gametophyt; Modell beruht auf botanischen Daten von de Boer; s. Prusinkiewicz & Lindenmayer 1990)

simulierte Entwicklung:

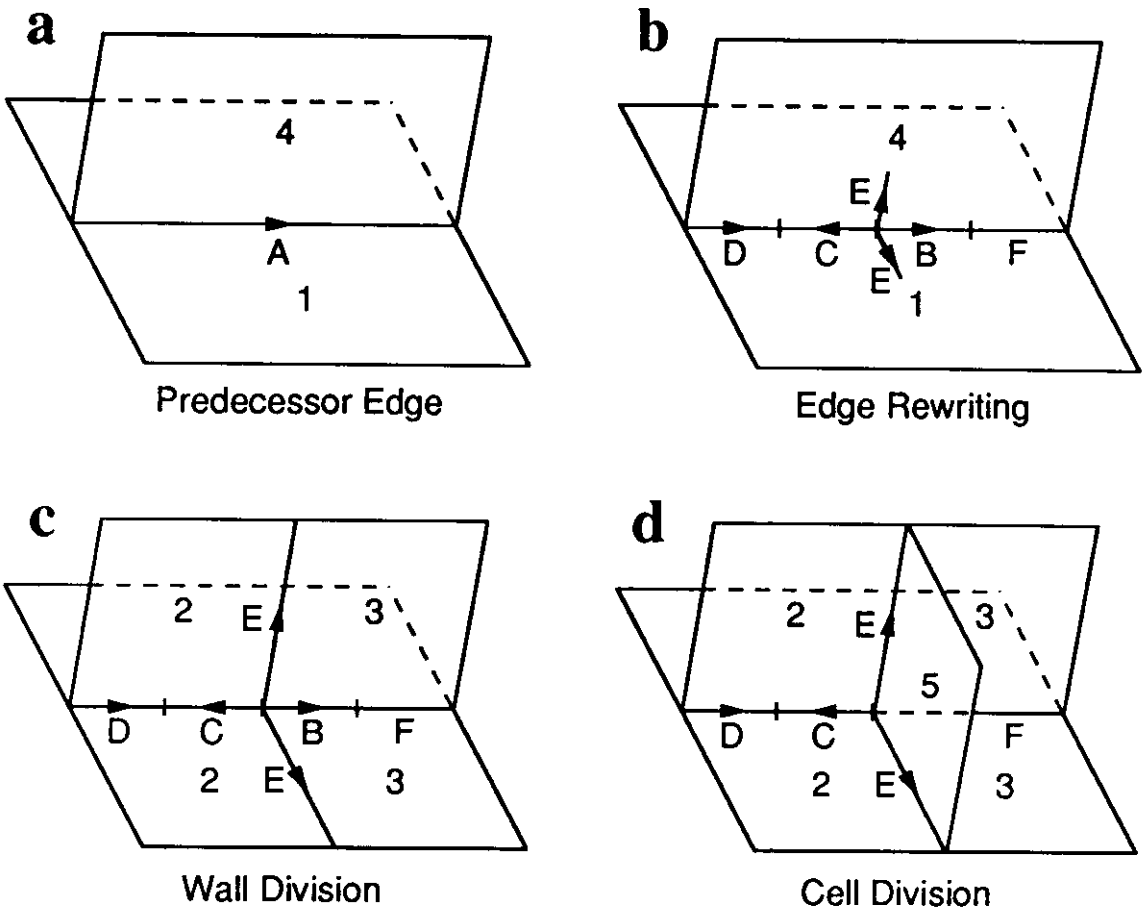


Mikrofotografie der realen Organismen:

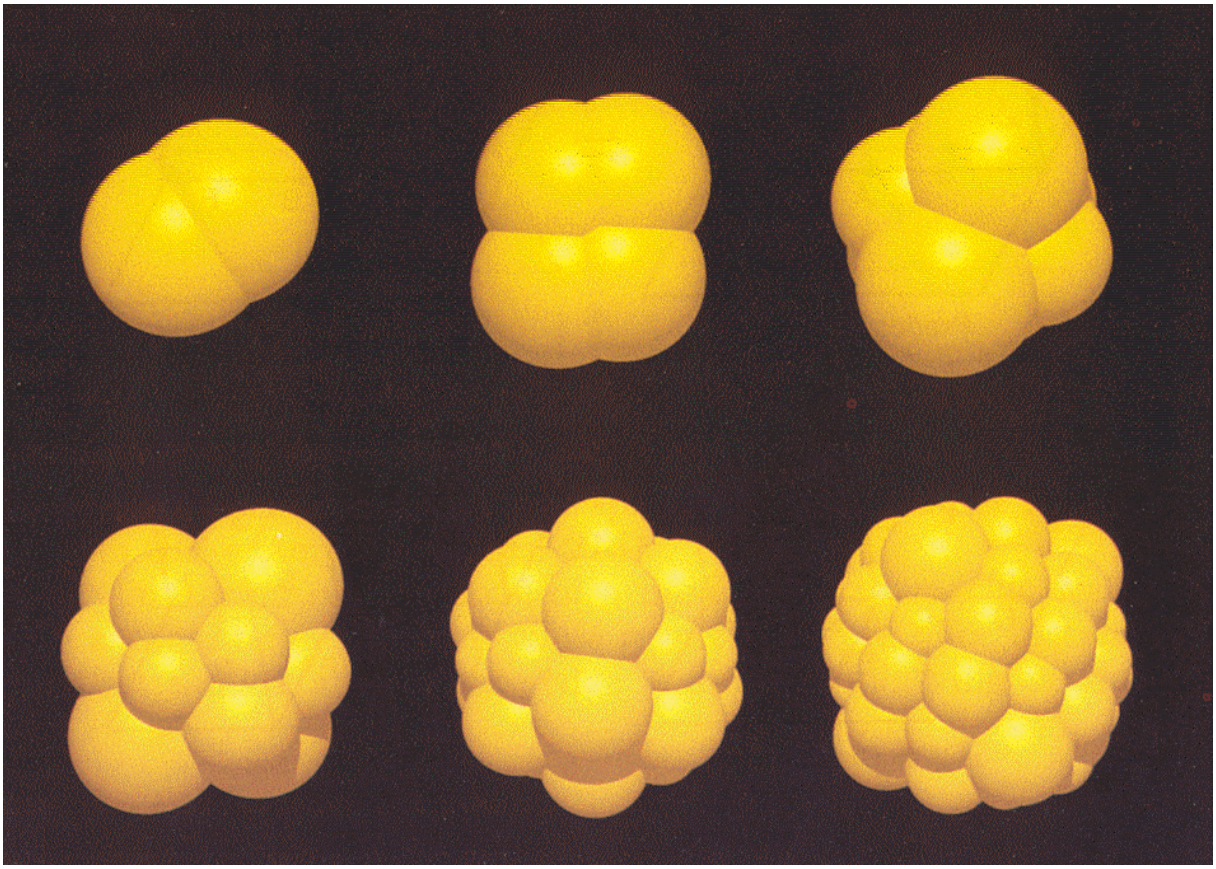


3D-Variante: *Cellwork L-systems*

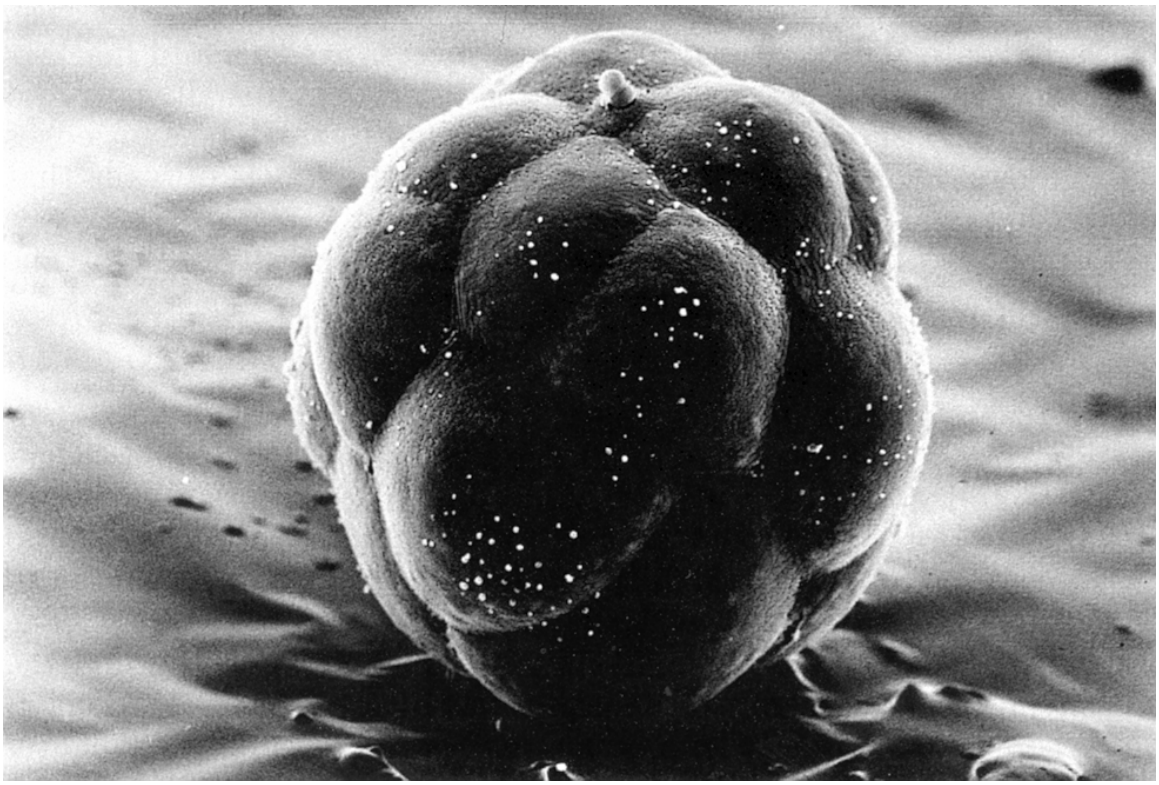
- Produktionsregeln wirken ebenfalls auf die Kanten
- 3 Phasen: Rewriting, Teilung der Zellwände, Teilung der Zellen (jeweils muss *matching* geprüft werden)



Beispiel: Modell des Schnecken-Embryos *Patella vulgata*



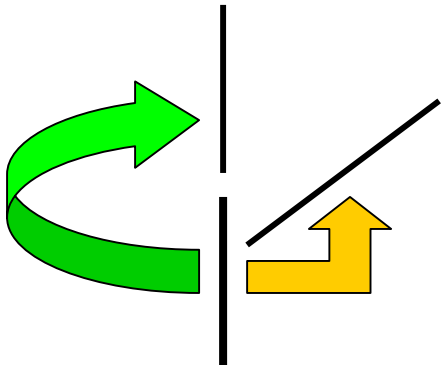
zum Vergleich: elektronenmikroskopische Aufnahme des realen Embryos



Relationale Wachstumsgrammatiken

als Erweiterung von L-Systemen

In L-Systemen mit Verzweigungen (über Turtle-Kommandos)
nur 2 mögliche Relationen zwischen Objekten:
"direkter Nachfolger" und "Verzweigung"



Erweiterungen:

- Zulassen weiterer Relationstypen (beliebig wählbar)
- Zulassen von Zyklen (→ Graph-Grammatik)
- Grammatik modifiziert direkt den Graphen, Umweg über String-Codierung entfällt (bzw. wird nur noch für Regel-Input gebraucht)

"*relationale Wachstumsgrammatik*"

außerdem Nachteil der Turtle-Interpretation von L-Systemen:
Segmente sind nur Zylinder, keine Objekte i. Sinne der OOP

Erweiterungen:

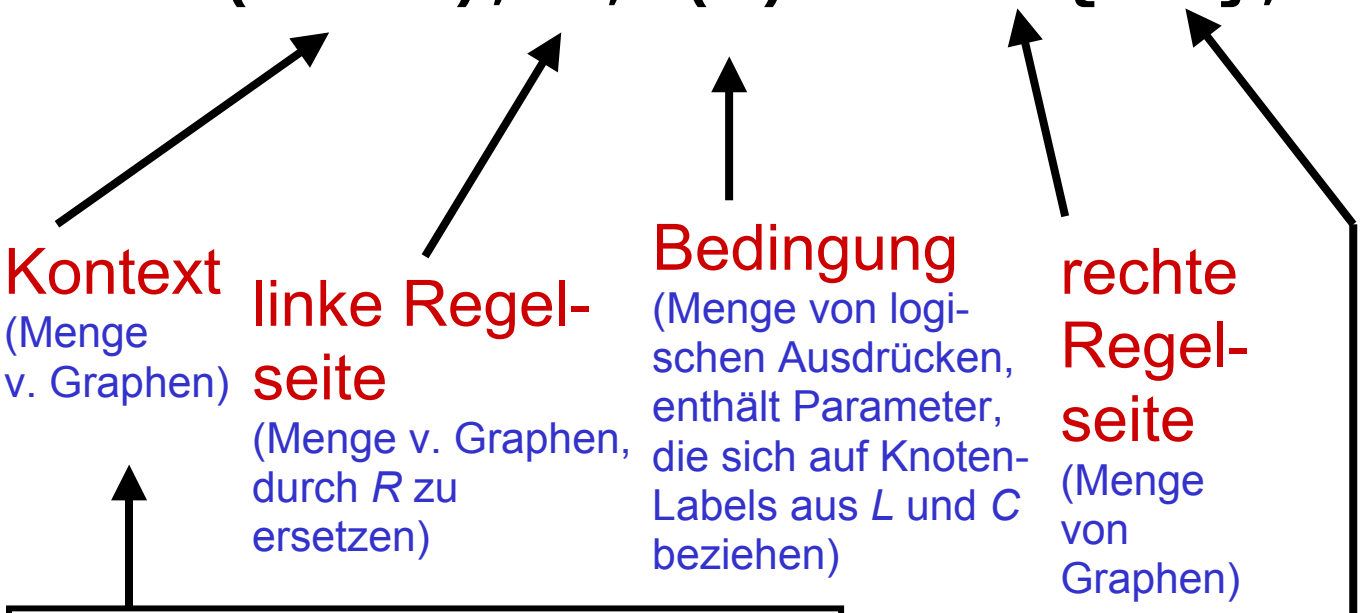
- Knoten des Graphen können beliebige Objekte sein (auch Grafikobjekte)
- Einbettung von Code einer höheren, imperativen und/oder objektorientierten Programmiersprache in die Regeln

Definition einer Regel einer relationalen Wachstumsgrammatik (Relational Growth Grammar, RGG):

(L, C, E, R, P) mit $L \cup C \neq \emptyset$

wir schreiben:

```
(* C *), L, (E) ==> R { P };
```



Kontext (Menge v. Graphen)

linke Regelseite (Menge v. Graphen, durch *R* zu ersetzen)

Bedingung (Menge von logischen Ausdrücken, enthält Parameter, die sich auf Knoten-Labels aus *L* und *C* beziehen)

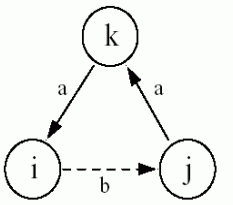
rechte Regelseite (Menge von Graphen)

prozeduraler Code (Liste von Kommandos)

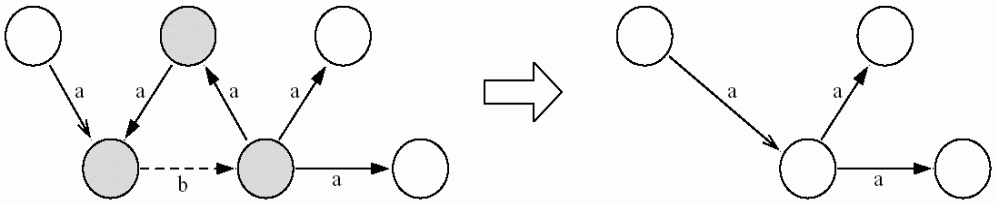
Graphen: gerichtet, mit Kanten- und Knoten-Labels

Beispiel:

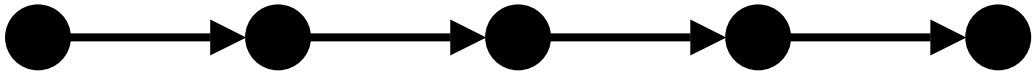
Regel



An-
wendung



RGGs als verallgemeinerte L-Systeme:
Strings sind spezielle Graphen



im Text werden allgemeine Kanten
geschrieben als **`-edgelabel->`**

Kanten des speziellen Typs "Nachfolger" werden
als Leerzeichen geschrieben
(anstatt **`-successor->`**)

Sprache XL (eXtended L-system language):

- RGG-Regeln in Blöcken organisiert
  → Kontrolle der Reihenfolge der Regelanwendungen
- Turtle-Kommandos als Knoten erlaubt
- Knoten sind Java-Objekte
- Sprache Java als Rahmen für die gesamte RGG
  → Benutzer kann Konstanten, Variablen, Klassen... definieren

Beispiel "Game of Life" als RGG:

zunächst Moore-Nachbarschaft als Relation definiert:

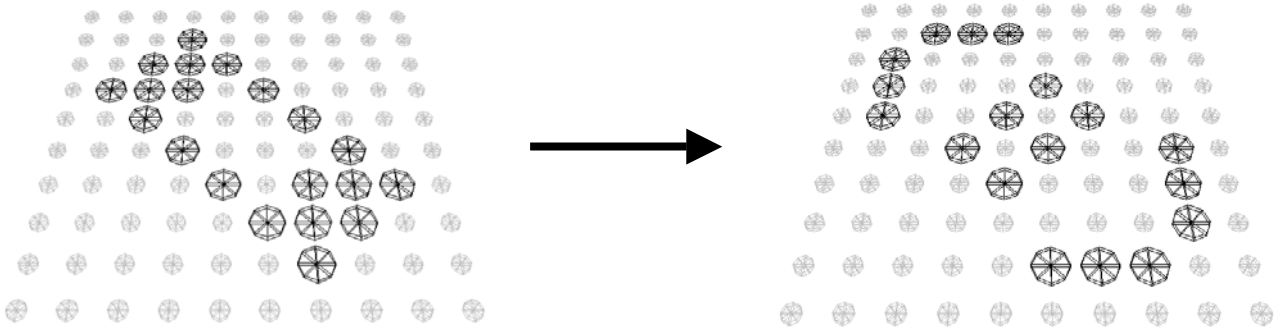
```
boolean neighbour(Cell c1, Cell c2)
  { return (c1 != c2) && (c1.distanceLinf(c2) < 1.1); }
```

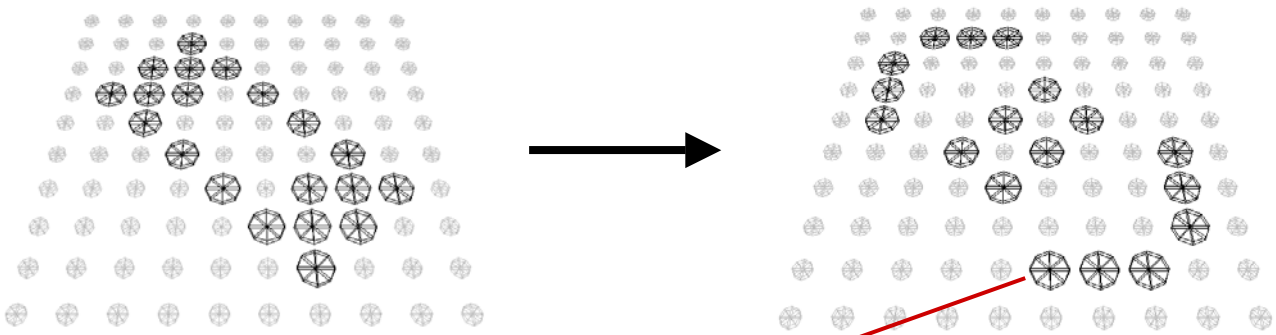
RGG-Regeln:

```
public void run()
  [  x:Cell(1), (!(sum((* x -neighbour-> #Cell *).state) in {2..3}))  ==> x(0);
     x:Cell(0), (  sum((* x -neighbour-> #Cell *).state) == 3)       ==> x(1);
  ]
```

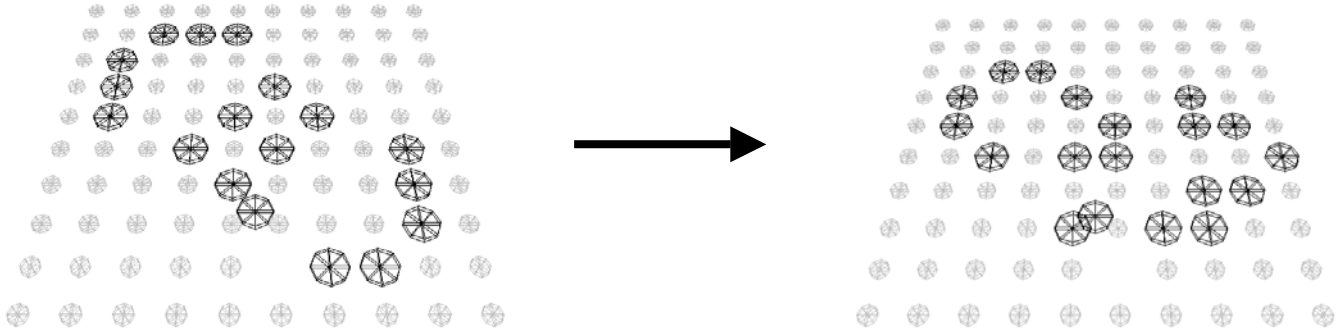
Visualisierung der Ergebnisse mit GroIMP (Growth grammar related Interactive Modelling Platform):

symmetrische Konfiguration,
ungestörte Entwicklung





Intervention des Benutzers,
führt zu Störung des Gitters und der Interaktion:



Verwendung auch für "realistische" Pflanzenmodelle:

Beispiel Gerste

das Modell verwendet im Hintergrund ein Netzwerk der Gibberellinsäure-synthese (Wachstums-hormon) zur Steuerung der Längen

laufendes Forschungs-vorhaben am Lehrstuhl (O. Kniemeyer, G. Buck-Sorlin)