

Lindenmayer Systeme

Seminar Bioinformatik
Wintersemester 2001

Manuel Klimek¹

¹email: klimek@fmi.uni-passau.de

1 Lindenmayer Systeme im Überblick	3
2 Sprachtheoretische Grundlagen	3
2.1 OL Systeme und deren Eigenschaften	3
2.1.1 OL, POL und DOL Systeme	3
2.1.2 Adult Sets von OL Systemen	4
2.1.3 EOL Systeme	4
2.1.4 Table OL Systeme	4
2.2 IL Systeme	5
2.3 Wachstumsfunktion	5
3 Anwendungsbeispiele in der Praxis	5
3.1 Lineare Interpretation	6
3.2 Bracketed L Systems	6
3.3 Turtle-Grafik	7
3.4 Map OL Systems	7

1 Lindenmayer Systeme im Überblick

2 Sprachtheoretische Grundlagen

2.1 0L Systeme und deren Eigenschaften

Formale Grundlage für 0L Systeme sind sog. *endliche Substitutionen* (Aus [?]).

Definition: Eine endliche Substitution ist eine Funktion aus Σ^* in die Menge aller endlichen, nicht-leeren Sprachen, die wie folgt definiert ist:

1. Für jeden Buchstaben $a \in \Sigma$ ist $\sigma(a)$ eine endliche, nicht-leere Sprache.
2. $\sigma(\lambda) = \lambda$.
3. Für alle Worte $w_1, w_2 \in \Sigma^*$ gilt $\sigma(w_1 w_2) = \sigma(w_1) \sigma(w_2)$.

Die Substitution heißt λ -frei, falls keine der Sprachen $\sigma(a)$, $a \in \Sigma$ das leere Wort enthält. Falls jedes $\sigma(a)$ nur aus einem Wort besteht, wird σ ein Morphismus genannt.

Eigenschaft (3) legt fest, dass die Ersetzung voll parallel in beliebiger Reihenfolge vorgenommen wird. Außerdem wird hier nicht, wie bei Phrase-Structure Grammatiken, zwischen Terminal- und Nichtterminalsymbolen unterschieden. Daraus ergeben sich grundlegende Unterschiede zwischen den erreichbaren Sprachen, die im folgenden noch eingehend diskutiert werden.

2.1.1 0L, P0L und D0L Systeme

Definition: (Aus [?]) Ein 0L System ist ein Tripel $G = (\Sigma, \sigma, w_0)$, wobei Σ ein Alphabet, σ eine endliche Substitution auf Σ und w_0 ein Wort aus Σ ist (w_0 heißt auch Axiom). Ein 0L System erzeugt die Sprache

$$L(G) = \{w_0\} \cup \sigma(w_0) \cup \sigma(\sigma(w_0)) \cup \dots = \bigcup_{i \geq 0} \sigma^i(w_0).$$

Eigenschaften von 0L Systemen:

- Die 0 im Namen **0L** System bedeutet, dass ein Zeichen jeweils unabhängig von seinen Nachbarn betrachtet wird. Dies ist eine primäre Eigenschaft von endlichen Substitutionen.
- Ein 0L System heißt **P0L** (propagierendes 0L) System, falls σ λ -frei ist. Das bedeutet, dass keine Buchstaben "verschwinden" können.
- Falls σ ein Morphismus ist, heißt das System **D0L** (deterministisches 0L) System. In diesem Fall gibt es für jeden Buchstaben des Alphabets genau eine Ableitungsmöglichkeit.

Man betrachte das PD0L System $(\{a\}, a \rightarrow a^2, a)$. Es erzeugt die Sprache $\{a^{2^n} \mid n \geq 0\}$. Diese Sprache ist nicht kontextfrei. Ein sehr einfaches L System kann also durch die parallele Ersetzung Sprachen erzeugen, die sich mit einfachen Phrase Structure Grammatiken nicht erzeugen lassen. Um mit 0L Systemen Linearität zu erreichen, verwendet man die Regel $a \rightarrow a$.

So kann man zum Beispiel die Sprache $\{a^n\}$ mit dem nichtdeterministischen POL System $(\{a\}, \{a \rightarrow a, a \rightarrow aa\}, a)$ erzeugen. Diese Sprache kann *nicht* von einem DOL System dargestellt werden. Aber es lassen sich nicht alle kontextfreien Sprachen mit OL Systemen darstellen. Die endliche Sprache $\{a, a^2\}$ kann zum Beispiel von keine OL System erzeugt werden.

2.1.2 Adult Sets von OL Systemen

Die Eigenschaft, dass sich nicht alle kontextfreien Sprachen mit OL Systemen erzeugen lassen rührt daher, dass bei OL Systemen keine Nichtterminale existieren. Jedes abgeleitete Wort ist automatisch in der Sprache. Adult Sets bezeichnen nun eine *Teilmenge* der erzeugten Worte und definieren somit selbst wieder eine Sprache. Da hier nur Worte *ausgelassen* werden, nennt man solche Erweiterungen der OL Systeme auch *Filter*.

Genauer ist ein Adult Set eines OL Systems die Menge aller der aus dem Axiom erzeugten Worte, die nur sich selbst erzeugen (aus [?, Seite 11 ff]). Es läßt sich beweisen, dass die Klasse der kontextfreien Sprachen gleich der Klasse der Adult Sets von OL Systemen ist. Die Möglichkeit nur einen Teil der Erzeugten Worte zur erzeugten Sprache hinzuzunehmen, ändert also die Eigenschaften des Systems grundlegen.

2.1.3 EOL Systeme

Wie Adult Sets sind auch Extended OL Systeme (EOL) Filter. Der Sprache werden, wie bei den Grammatiken der Chomsky Hierarchie, Nichtterminalsymbole hinzugefügt. Worte, die eines dieser Symbole beinhalten, sind dann nicht in der erzeugten Sprache. Kombiniert mit der Möglichkeit Terminalsymbole zu ersetzen, kann man mit diesem System auch kontextsensitive und nicht kontextfreie Sprachen, wie z.B. $a^n b^n c^n$ erzeugen. Man betrachte das EOL System mit dem Axiom ABC und folgenden Projektionen:

$$\begin{array}{l} A \rightarrow AA' \quad A \rightarrow a \quad A' \rightarrow A' \quad A' \rightarrow a \quad a \rightarrow F \\ B \rightarrow BB' \quad B \rightarrow b \quad B' \rightarrow B' \quad B' \rightarrow b \quad b \rightarrow F \\ C \rightarrow CC' \quad C \rightarrow c \quad C' \rightarrow C' \quad C' \rightarrow c \quad c \rightarrow F \\ F \rightarrow F \end{array}$$

Da jedes Terminalsymbol sobald es einmal erzeugt wurde sofort in das Nichtterminal F übergeht, müssen alle Terminalsymbole gleichzeitig erzeugt werden, damit das Wort in der erzeugten Sprache ist. Das System ist also *synchronisiert* [?, Seite 264, 265].

2.1.4 Table OL Systeme

Zur Darsetzung von OL Systemen, deren Entwicklung sich im Laufe der Zeit verändert, gibt es sogenannte TOL (Table OL) Systeme.

Definition: (Aus[?]) Ein TOL System ist ein Tripel $G = (\Sigma, S, w_0)$, wobei S eine endliche Menge endlicher Substitutionen ist, so dass für jedes $\sigma \in S$ das Tripel (Σ, σ, w_0) ein OL System ist. Die von einem TOL System erzeugte Sprache $L(G)$ besteht aus w_0 und allen Worten in allen Sprachen $\sigma_1 \dots \sigma_k(w_0)$, wobei $k \geq 1$ und jedes $\sigma_i \in S$. Sind alle Substitutionen in S Morphismen, so ist G ein deterministisches TOL System (DTOL).

2.2 IL Systeme

Die Bezeichnung IL Systeme steht für L Systeme, bei denen Interaktion mit den Nachbarzellen (Buchstaben) stattfindet. Man schreibt (m, n) L System, falls die Regeln m Buchstaben links und n Buchstaben rechts des zu ersetzenden Buchstabens berücksichtigen. Im Gegensatz zu kontextsensitiven Sprachen wird hier nur *ein* Buchstabe ersetzt, jedoch abhängig von seinen Vorgängern und Nachfolgern. Die Ersetzungsregeln haben nun folgende Form [?, Seite 272, 273]:

$$(\alpha, a, \beta) \rightarrow w, |\alpha| = m, |\beta| = n.$$

Um immer eine genügend große Anzahl von Buchstaben zur Verfügung zu haben, denkt man sich das Wort umhüllt von dem sogenannten Umgebungssymbol #. Auch für dieses Symbol müssen natürlich Regeln vorhanden sein.

2.3 Wachstumsfunktion

Die *Wachstumsfunktion*¹ $f(n)$ eines deterministischen (oder stochastischen) L Systems ist die (erwartete) Wortlänge des n -ten generierten Wortes. Da es für das Wachstum unerheblich ist, in welcher Reihenfolge die Buchstaben in einem Wort vorkommen, reicht es, sich nur die *Anzahl* jedes Buchstaben in einem Wort zu merken. Diese wird mit Hilfe eines Vektor π , dem *Parikh Vektor*, dargestellt. Die *Wachstumsmatrix* M des L Systems speichert nun in jeder Reihe für einen Buchstaben die Anzahl der nach der Ersetzung aus diesem Buchstaben entstehenden Buchstaben derart, dass $\pi_n M = \pi_{n+1}$. Bezeichnen wir mit μ den konstanten 1-Vektor, so ist

$$f(n) = \pi M^n \mu$$

Eigenschaften von Wachstumsfunktion:

- Die Wachstumsfunktion eines propagierenden L Systems ist monoton steigend.
- Ist die Wachstumsfunktion an einer Stelle 0, so ist sie an allen folgenden Stellen ebenfalls 0.
- Hat man die Wachstumsmatrix einer Wachstumsfunktion berechnet, so kann man die Wachstumsfunktion auch explizit berechnen [?],[?].

3 Anwendungsbeispiele in der Praxis

Um mit L Systemen reale Systeme modellieren zu können, muss man ein eine *Interpretation* der erzeugten Worte festlegen und sich für eine das Problem unter der Interpretation möglichst genau abbildende Variante der L Systeme entscheiden (*Inferenzproblem*²). Im folgenden werden nun einige Beispiele zur Lösung dieser Probleme behandelt.

¹vgl. [?, Seite 15ff], [?, Seite 288ff]

²vgl.[?, Seite 9f]

3.1 Lineare Interpretation

Die wohl einfachste biologische Interpretationsweise ist, jeden Buchstaben als Zelle eines Organismus zu interpretieren. Auf diese Weise lassen sich verschiedene Algen, sowie auch die Bakterie *Anabaene catenula* darstellen [?], [?]. Hier symbolisieren die Buchstaben a und b den Zellstatus (Größe und Teilbereitschaft der Zelle). l und r legen die Polarität der Zelle fest, d.h. auf welcher Seite der Zelle eine Tochterzelle entsteht. Es ist also $\Sigma = a_l, a_r, b_l, b_r$. Weiter sei $w_0 = b_r$ und σ definiert durch:

$$a_r \rightarrow a_l b_r$$

$$a_l \rightarrow b_l a_r$$

$$b_r \rightarrow a_r$$

$$b_l \rightarrow a_l$$

Die daraus folgende Sequenz

$$b_r$$

$$a_r$$

$$a_l b_r$$

$$b_l a_r a_r$$

$$a_l a_l b_r a_l b_r$$

...

modelliert die Entwicklung der *Anabaene catenula*.

3.2 Bracketed L Systems

Da viele Organismen nicht mit einer linearen Aneinanderreihung von Zellen dargestellt werden können, benötigt man eine Erweiterung der Interpretation. Man führt dazu entsprechende Verzweigungssymbole ein, die in das Alphabet aufgenommen werden.

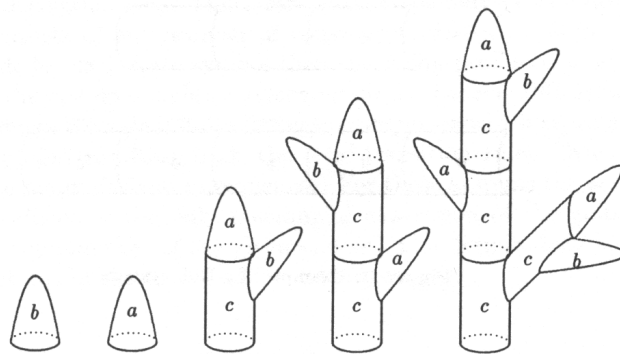


Abbildung 1: Einfache, verzweigte Strukturen

Die in Abbildung 1 dargestellte Entwicklung kann ein DOL System mit dem Axiom b und folgenden Produktionen erreicht werden [?]:

$$a \rightarrow c[b]a, \quad b \rightarrow a, \quad c \rightarrow c, \quad [\rightarrow [,] \rightarrow],$$

Nach Lindenmayer und Jürgensen [?], weisen unter den einfachen Pflanzen sehr viele solche Verzweigungsstrukturen auf. Ein Beispiel ist die *Callithamnion roseum*.

3.3 Turtle-Grafik

Um nicht nur die relativ abstrakte Verzweigung der Teilstücke, sondern auch deren genau Position im Raum darstellen zu können, interpretiert man die Buchstaben nun nicht mehr als Zellen, sondern als Anweisungen, um ein virtuelles Turtle durch den Raum zu fahren und dabei Linien zu ziehen. Die Anweisungen kann man dabei zum Beispiel wie folgt festlegen [?, Seite 19ff]:

F Gehe um d vorwärts und zeichne dabei eine Linie.

f Gehe um d vorwärts ohne zu zeichnen.

+ Drehe um den Winkel δ nach links.

– Drehe um den Winkel δ nach rechts.

[Merke den Turtle-Zustand (Position und Richtung) auf dem Stack.

] Hole den Turtle-Zustand vom Stack.

Der Winkel δ und die Schrittlänge d sind Turtle-Variablen und werden vor dem Zeichnen einmal festgelegt.

Die *Kantenersetzung* ist eine der zwei Möglichkeiten, wie die Graphen eines Turtles entstehen können. Dabei wird ein Schritt F des Turtles durch eine Folge von Turtlebefehlen ersetzt. Will man komplexere Strukturen ermöglichen, kann man mehrere Symbole für eine Kante einführen, die in unterschiedlichen Ersetzungsregeln resultieren [?, Seite 11ff].

Knotenersetzung ist die andere Art, Turtle-Grafiken zu bilden. Hier wird für einen Knoten ein neues Symbol eingeführt, welches beim Zeichnen des Turtles meist ignoriert wird.

Mit Hilfe dieser Methoden können schon einfache, pflanzenähnliche Strukturen erzeugt werden. Abbildung 2 zeigt links eine durch Kantenersetzung, rechts eine durch Knotenersetzung erzeugte Turtle-Grafik. Sind die erzeugenden L Systeme nichtdeterministisch, so kann man, indem man in jedem Ersetzungsschritt zufällig eine der möglichen Regeln für einen Buchstaben auswählt, die "zufällige" Entwicklung realer Pflanzen simulieren.

3.4 Map OL Systems

Map L Systems werden zur Modellierung zellulärer Strukturen verwendet. Eine Map ist eine endliche Menge von *Regionen*. Jede Region wird von einer endlichen Sequenz zusammenhängender Kanten umgeben, die sich an Knotenpunkten berühren. Die Menge der Kanten ist dabei verbunden, d.h. es gibt einen Pfad von jedem Knoten zu jedem anderen Knoten und jede Kante ist mit mindestens einer anderen Kante verbunden [?].

Ein mBPMOL-System (Binary Propagating Map OL System With Markers [?]) wird unter anderem von Tuzza und Linenmayer [?] verwendet, um Wachstum und Art von Zellen der Retina zu beschreiben.

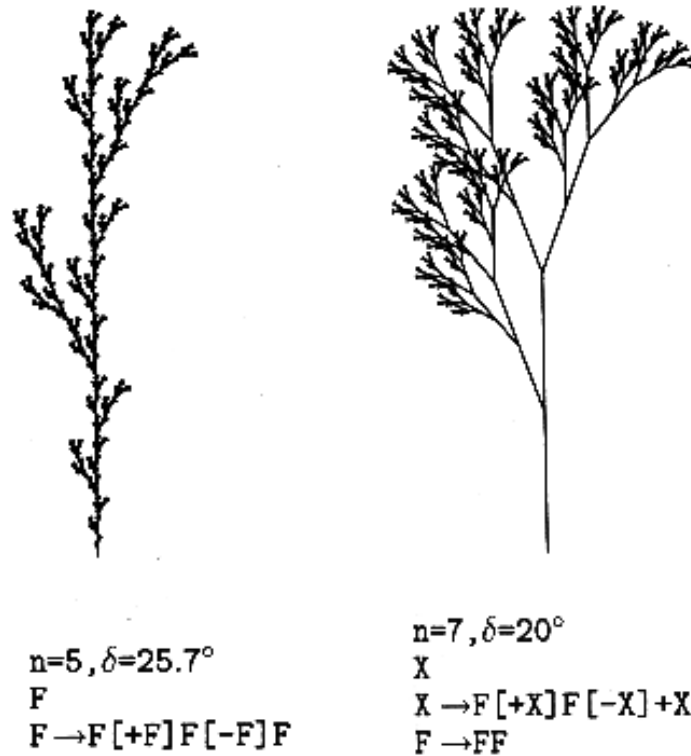


Abbildung 2: [?, Seite 25]

Literatur

- [1] Salomaa A. Kari L., Rozenberg G. *Handbook of Formal Languages*, volume 1, chapter 5. Springer, Berlin, 1997.
- [2] Jürgensen H. Lindenmayer A. Grammars of development: Discrete-state models for growth, differentiation, and gene expression in modular organisms. In Salomaa A. Rozenberg G., editor, *Lindenmayer Systems*. Springer, Berlin, 1992.
- [3] Tuza Z. Lindenmayer A. Locally generated colourings of hexagonal cell division patterns: Application to retinal cell differentiation. In Salomaa A. Rozenberg G., editor, *Lindenmayer Systems*. Springer, Berlin, 1992.
- [4] P. Prusinkiewicz. *The Algorithmic Beauty Of Plants*. Springer, Berlin, 1990.