Lindenmayer-Systeme: Pflanzenwachstum als formale Sprache im Kontext der Computergrafik

Raphael Wudy
Fakultät für Informatik
Technische Hochschule Rosenheim
Rosenheim, Deutschland
raphael.wudy@stud.th-rosenheim.de

***Zusammenfassung*—Diese Arbeit untersucht die Verbindung zwischen der Chomsky-Hierarchie und Lindenmayer-Systemen (L-Systemen) sowie deren Anwendung in der Computergrafik. Die Chomsky-Hierarchie bietet ein Klassifikationsschema für formale Sprachen, während L-Systeme ein Modell zur Beschreibung biologischer Wachstumsprozesse sind. Zuerst erläutert diese Arbeit die theoretischen Grundlagen und versucht die L-Systeme in die Chomsky-Hierarchie einzuordnen und zwar durch einen Vergleich der Eigenschaften und Produktionsregeln. Nach der Einordnung geben praktische Beispiel einen Einblick in die Visualisierung der L-Systeme, welche mithilfe der Turtle Interpretation umgesetzt ist. Die anschließende Diskussion bezieht sich auf die Möglichkeiten und Grenzen der L-Systeme in der Computergrafik. Dabei stellte sich heraus, dass L-Systeme zwar die zu zugrundeliegende formalen Sprachkonzepte verwenden, allerdings sich nicht in die von Noam Chomsky kreierte Hierarchie vollständig eingliedern lassen. Die 0L-Variante der L-Systeme ähnelt dem Sprachtyp-2. Dies ist darauf zurückzuführen, dass die Regeln zur Produktion, Einschränkungen und der Kontext gleich sind, wobei L-Systeme die Produktionen parallel anwenden und auf Terminale verzichten. Nichtsdestotrotz kann eine Terminierung der generierten Zeichenkette implementiert werden. Außerdem existieren Limitationen in L-Systemen, welche den Anwendungsbereich dieser einschränken. Sie generieren zuverlässige 3D-Modelle einer Pflanze ohne Berücksichtigung physikalischer Eigenschaften. Im Grunde eigenen sich L-Systeme zur Modellierung von Pflanzen und Wachstumsprozessen und sind eher im wissenschaftlichen Bereich angesiedelt und gelten nicht als Industrie-Standard. L-Systeme dienen als Grundlage für einen Compiler bei prozedural generierten Strukturen oder bei Pflanzenwachstumsprozessen.**

***Index Terms*—Chomsky-Hierarchie, L-Systeme, formale Sprachen, Computergrafik, Pflanzenmodellierung, generative Algorithmen**

I. Einleitung

Der Sprachwissenschaftler Noam Chomsky entwickelte in den 1950er Jahren eine Klassifikation formaler Sprachen, die als Chomsky-Hierarchie bekannt ist. Diese bildet bis heute eine Grundlage der theoretischen Informatik und findet insbesondere Anwendung im Compilerbau und in der Sprachverarbeitung. Parallel dazu stellte der Botaniker Aristid Lindenmayer 1968 ein Modell zur Beschreibung biologischer Wachstumsprozesse vor: die Lindenmayer-Systeme (L-Systeme) [1]. Während die Chomsky-Hierarchie primär auf Sprachstrukturen abzielt, fanden L-Systeme aufgrund ihrer Ähnlichkeit zu formalen Grammatiken auch Anwendung in der Informatik, insbesondere in der Computergrafik zur Modellierung von Pflanzen.

Zunächst gab es keine Hinweise auf die Verwendung von L-Systemen in der grafischen Darstellung bis in die 1980er Jahre [2]. Die L-Systeme zeigen Ähnlichkeiten im Aufbau zu formalen Sprachen, weshalb die Chomsky-Hierarchie in Betracht zu ziehen ist. Des Weiteren stellt sich die Frage, wie Zeichenketten, die L-Systeme generieren, grafisch zu interpretieren sind.

II. Fragestellung und Methodik

Eine Literaturrecherche mit Dokumentenanalyse bildet die Grundlagen für eine theoretische Untersuchung und die praktischen Beispiele. Zunächst werden relevante Quellen zu L-Systemen und der Chomsky-Hierarchie gesammelt und ausgewertet. Im Rahmen der Dokumentenanalyse vergleicht die Arbeit Produktionsregeln sowie Eigenschaften von L-Systemen und den Sprachtypen der Chomsky-Hierarchie. Eine Beweisführung soll klären, ob und wie L-Systeme in der Hierarchie regulärer, kontextfreier, -sensitiver oder rekursiv aufzählbaren Sprachen einzuordnen sind und welche Vorteile die Einordnung mit sich bringt.

Das Heranziehen von praktischen Beispielen soll der Veranschaulichung der theoretischen Grundlagen dienen. Diese Arbeit geht dabei tiefer auf die Verwendung von L-Systemen in der Computergrafik und -gestützten Botanik ein. Im Vordergrund steht hier die Modellierung von Wachstumsprozessen von Pflanzen. Eine abschließende Diskussion führt die Ergebnisse zusammen und interpretiert diese. Abgerundet mit einem Fazit der gewonnenen Erkenntnisse.

Die Arbeit soll dabei klären, in welchem Umfang L-Systeme eine formale Sprache zur Beschreibung von Wachstumsprozessen einer Pflanzen sind und welche grafischen Interpretationsmöglichkeiten bestehen.

III. Theoretische Grundlagen

A. Formale Sprachen und Grammatik

Eine formale Sprache ist ein mathematisches System. Dadurch können strenge Aussagen über formale Sprachen entstehen. Die formale Sprache setzt sich aus einer endlichen Anzahl von Symbolen (Alphabet, Σ), bspw. das binäre Alphabet = $\{0, 1\}$, und einem Sprachschatz (Wort), bestehend aus einer endlichen Anzahl an kombinierten Symbolen des Alphabets, bspw. Wort = $\{0, 1, 001, 101\}$, zusammen [3]. Dabei ist die

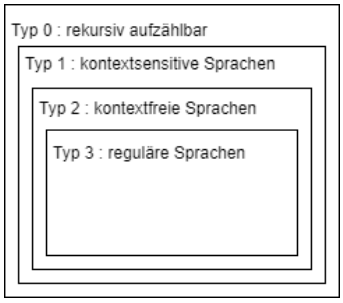


Abbildung 1. visuelle Darstellung der Chomsky-Hierarchie

Grammatik (G) ein Werkzeug zur Erzeugung und Validierung der formalen Sprache. G ist definiert als ein Quadrupel (V, T, P, S) und repräsentieren Variablen, Terminale, Produktionen und den Start. Produktionen haben die Form $\alpha \rightarrow \beta$ [3].

Die formale Sprache L, generiert durch die Grammatik G, ist nach der Form von Gleichung 1 [3] zu definieren.

$$L(G) = \{w | w \in T^* \text{ und } S \overset{*}{\underset{G}{\rightarrow}} w\} \tag{1}$$

Beispiel: Grammatik G = (V, T, P, S), wobei V = {S}, T = {0,1}, P = {S $\rightarrow$ 0S1, S $\rightarrow$ 01}. generiert die formale Sprache L(G) = $\{0^n 1^n | n \geq 1\}$.

Formale Sprachen bilden dabei in der Informatik ein zentrales Konzept in verschiedenen Teilbereichen. In der Berechnungstheorie dienen sie zur Analyse von Algorithmen und Rechenmodellen wie Turing-Maschinen oder endliche Automaten [4]. Im Compilerbau erlauben sie lexikalische und Syntaxanalyse, sowie Codegenerierung [5]. Im Bereich der Computergrafik finden sie ihren Platz bei der prozeduralen Generierung oder Simulation [6].

B. Chomsky-Hierarchie

In seiner Theorie zu Sprachen und deren Grammatiken entwickelte Noam Chomsky eine hierarchische Klassifizierung. Die Bedingungen und Strukturen eines jeden Sprachtyps unterscheiden sie eindeutig von einander, wodurch eine Abgrenzung in die sogenannte Chomsky-Hierarchie ermöglicht wird. Trotz dieser Hierarchie und Abgrenzung bauen die Sprachtypen aufeinander auf und bildet echte Teilmengen. Es gilt: Typ-3-Sprachen $\subset$ Typ-2-Sprachen $\subset$ Typ-1-Sprachen $\subset$ Typ-0-Sprachen (vgl. Abbildung 1) [7], [8].

1) ***Typ-3: Reguläre Sprachen****:* Zur Erkennung dieser Sprachen existiert ein endlichen Automat (M), sodass nach Einlesen des letzten Symbols einer Zeichenkette $w \in$ L(G) der Zustand von M $\in$ T(M) ist. Wobei T(M) die Menge aller Endzustände von M ist. Neben dieser Tatsache sind Typ-3 Sprachen abgeschlossen unter den Operationen: [3], [8]:

- **Vereinigung:** $L_1 \cup L_2$
- **Konkatenation:** $L_1 L_2$
- **Stern-Operation (Kleene-Stern):** L^*
- **Komplementbildung:** $\overline{L}$
- **Schnitt:** $L_1 \cap L_2$

Produktionen sind von der Form A $\rightarrow$ aB | A $\rightarrow$ Ba | A $\rightarrow$ ab und damit rechts- bzw. linkslinear.

Beispiel: Sei G = ({S, B}, {0,1}, P, S), mit P = $\{S \rightarrow 0B, B \rightarrow 0B, B \rightarrow 1B, B \rightarrow 0\}$. Daraus folgt nach formaler Sprachtheorie die Sprache:
L(G) = { w $\in \{0,1\}^* | w$ beginnend und endend mit einer 0}. Diese Typ-3 Sprache ist rechtslinear. Ein endlicher Automat zur Erkennung von Wörtern der Sprache L(G) ist in Abbildung 2 zusehen.

Reguläre Sprachen bilden die unterste Stufe in der Chomsky-Hierarchie und finden Verwendung im Compilerbau [3], [8].

2) ***Typ-2: Kontextfreie Sprachen****:* Die Grammatiken dieser Typen weisen die Form A $\rightarrow \gamma$, mit A $\in$ V und $\gamma \in$ (V $\cup\Sigma)^*$. Kellerautomaten bzw. Pushdown Automaten erkennen kontextfreie Sprachen [3], [8].

Typ-2 ist abgeschlossen unter [3]:

- **Vereinigung:** $L_1 \cup L_2$
- **Konkatenation:** $L_1 L_2$
- **Stern-Operation (Kleene-Stern):** L^*

und gilt als nicht abgeschlossen unter [3]:

- **Komplementbildung:** $\overline{L}$
- **Schnitt:** $L_1 \cap L_2$

Beispiel: Gegeben ist die Grammatik G = ({S}, {a,b}, P, S), wobei P = {S $\rightarrow$ aSb, S $\rightarrow$ ab}. Diese Grammatik erzeugt die Sprache L(G) = $\{a^n b^n | n \geq 1\}$. Dieses Beispiel zeigt, dass es kontextfreie Sprachen gibt, die nicht dem Sprachtyp-3 angehören und dadurch eine eigene Klassifizierung bilden. Wie eingangs erwähnt gilt Typ-3-Sprachen $\subset$ Typ-2-Sprachen [7].

3) ***Typ-1: Kontextsensitive Sprachen****:* Die Grammatik von Typ-1-Sprachen hat die allgemeine Form $\alpha A \beta \rightarrow \alpha \gamma \beta$, wobei $A \in V, \quad \alpha, \beta \in (V \cup \Sigma)^*, \quad \gamma \in (V \cup \Sigma)^+$ und $|\gamma| \geq 1$ gilt. Deterministische linear-beschränkte-Automaten können kontextabhängig Sprachen erkennen, die eine höhere Ausdrucksmächtigkeit als Typ-2 Sprachen besitzen [3], [8]. Sie sind abgeschlossen unter den folgenden Operationen [3]:

- **Vereinigung:** $L_1 \cup L_2$
- **Konkatenation:** $L_1 L_2$
- **Stern-Operation (Kleene-Stern):** L^*
- **Komplementbildung:** $\overline{L}$
- **Schnitt:** $L_1 \cap L_2$

Beispiel: Gegeben sei die Grammatik G = ({S, A, B, C}, {a,b,c}, P, S), wobei P = $\{S \rightarrow aSBC | aBC, CB \rightarrow$

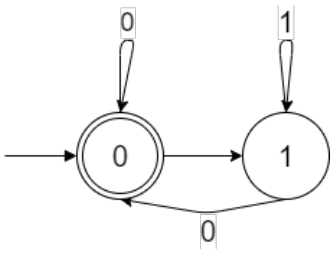


Abbildung 2. Endlicher Automat zur Erkennung der Sprache L(G) = { w $\in \{0,1\}^* | w$ beginnend und endend mit einer 0}

Generation	Zeichenkette
1	a
2	cbc
3	kdadk
4	kacbcak
5	kcbckdadkcbck
6	kkdadkkacbcakkdadkk
7	kkacbcakkcbckdadkcbckkacbcakk
8	kkcbckdadkcbckkkdadkkacbcakkdadkkkcbckdadkcbckk

Tabelle I
ZEICHENKETTEN NACH GENERATION IM L-SYSTEM

$BC, aB \rightarrow ab, bB \rightarrow bb, bC \rightarrow bc, cC \rightarrow cc\}$. Diese Grammatik erzeugt die Sprache $L = \{a^n b^n c^n \mid n \geq 1\}$ [9].

Wie zur Abgrenzung der Typ-2 Sprachen mit Typ-3 Sprachen, zeigt dieses Beispiel, dass Typ-2 Sprachen Ausdrucksmächtigkeit fehlt. Der Schnitt zweier Typ-2 Sprachen $a^n b^n$ mit $a^n c^n$ konstruiert die Sprache im angeführten Beispiel. Es verdeutlicht den Unterschied zwischen Typ-1 und Typ-2 Sprachen. Zusätzliche Abhängigkeiten und komplexere Produktionen erlauben dies den Typ-1 Sprachen [3], [7], [8].

4) Typ-0: rekursiv aufzählbar: Den letzten und mächtigsten Sprachtyp bilden Typ-0 Sprachen. Sie sind rekursiv aufzählbar und durch Turing-Maschinen erkennbar. Ihre Produktionen haben die allgemeine Form $\alpha \rightarrow \beta$, wobei $\alpha, \beta \in (V \cup \Sigma)^+$ und $\alpha \neq \epsilon$ gilt [3], [8].

Typ-0-Sprachen sind abgeschlossen unter den folgenden Operationen [3]:

- **Vereinigung:** $L_1 \cup L_2$
- **Konkatenation:** $L_1 L_2$
- **Stern-Operation (Kleene-Stern):** L^*
- **Komplementbildung:** $\overline{L}$
- **Schnitt:** $L_1 \cap L_2$

Beispiel: Gegeben sei die Grammatik $G = (\{S, A, B\}, \{a\}, P, S)$, wobei P = $\{S \rightarrow ACaB, Ca \rightarrow aaC, CB \rightarrow DB, CB \rightarrow E, aD \rightarrow Da, AD \rightarrow AC, aE \rightarrow Ea, AE \rightarrow \varepsilon\}$. G erzeugt die Sprache L(G) = $\{a^{2^n} \mid n \geq 0\}$ [9].

Mit dieser Mächtigkeit entstehen nicht entscheidbare Probleme, wie z. B. das Halteproblem. Typ-0 Sprachen umfassen alle anderen Sprachtypen und runden somit den Ausdruck Typ-3-Sprachen $\subset$ Typ-2-Sprachen $\subset$ Typ-1-Sprachen $\subset$ Typ-0-Sprachen [7] ab.

C. L-Systeme

Der Botaniker Aristid Lindenmayer entwickelte im Jahre 1968 den mathematischen Formalismus der L-Systeme. Die L-Systeme ermöglichen ihm die Modellierung von Zellteilung (und anderen zellularen Verhalten) in multizellularen Organismen darzustellen. Das zentrale Konzept hinter L-Systemen ist das Umschreiben. Sie ersetzen komplexe Objekte sukzessive mit einfacheren Objekten durch Anwendung von Umschreibungs- oder Produktionsregeln. Diese Regeln kann ein L-System parallel anwenden, da die Zellteilung in einem Organismus nicht sequenziell passiert, sondern zur selben Zeit an verschiedenen Stellen im Organismus. Sie ersetzen jedes Symbol in einem Wort gleichzeitig. Dieser mathematische Formalismus wurde speziell zur Anwendung in der computergesützten Botanik zur Simulation von Wachstumsprozessen entwickelt und baut auf Automatentheorie und den formalen Sprachen auf. Dabei entstanden verschiedene Varianten der L-Systeme wie z.B.das kontextfreie 0L-System, das kontextsensitive $\langle m, n \rangle$L-System bzw. parametrische L-Systeme und partielle L-System [1], [2], [6]. Diese Arbeit untersucht im nachfolgenden 0L-Systeme bzw. deterministische 0L-Systeme (D0L-Systeme) genauer. Eine Analyse anderer oder mehrerer Varianten würde den Umfang der Arbeit überschreiten.

*1) **0L-Systeme**:* Bestehend aus einem geordneten Tripel G = (V, ω, P), mit V als Alphabet des Systems, $\omega \in V^+$ als nicht leeres Wort (Axiom) und P $\subset V \times V^*$ ist ein endliches Set an Produktionen. Eine Produktion ist ein Tupel (a, χ) $\in$ P geschrieben als $a \rightarrow \chi$.

Existiert für den Buchstaben a keine Produktion in P, so wird die Identitätsproduktion ($a \rightarrow a$) angenommen. Ein 0L-System ist deterministisch (D0L-System), wenn für jedes $a \in V$ exakt ein $\chi \in V^*$ mit $a \rightarrow \chi$ existiert Ein 0L-System berücksichtigt dabei keinen Input zwischen den 'Zellen' und ist dadurch kontextfrei [1], [6].

Der Nutzen dieser Systeme für biologische Probleme besteht in erster Linie darin ein Verständnis der morphogenetischen Kraft von Zelllinien zu erlangen. Gemeint ist das Erkennen von Strukturen, die durch autonom programmierte Zellen entstehen können. Das Verhalten ist durch ihre Abstammung (Urzelle) kontrolliert [1].

Beispiel: Gegeben ist ein Tripel G = ({a,b,c,d,k}, a, P), wobei P = {(a $\rightarrow$ cbc), (b $\rightarrow$ dad), (c $\rightarrow$ k), (d $\rightarrow$ a), (k $\rightarrow$ k)}. Unter Anwendung der Produktionsregeln mit dem Axiom a entsteht ein String über mehrere Generationen hinweg (vgl. Tabelle I). Dieser generierte String beschreibt das Wachstum der Pflanze. Allerdings ist diese Repräsentation für den Menschen schwer zu interpretieren, weshalb Lindenmayer eine visuelle Repräsentation an die Variablen des Tripel koppelt und ihnen dadurch mehr Interpretationskraft zuspricht. In diesem Beispiel ist der erste Schritt zur visuellen Darstellung das festlegen der visuellen Eigenschaft pro Variable. a,b stellen

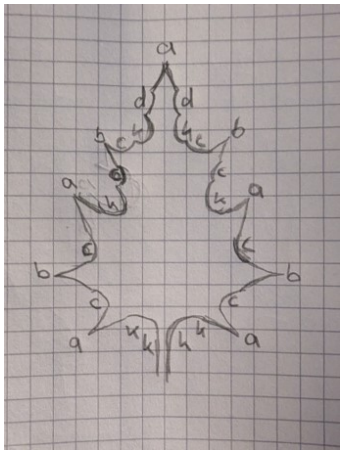


Abbildung 3. Manuelle Anwendung der grafischen Interpretation des 0L-Systems. Blatt einer Stachelpalme

ein spitz zulaufendes Dach dar. c,d eine leicht nach innen gewölbte Linie und k eine starke Vertiefung nach innen. Des Weiteren gilt, je mehr Variablen derselben Art hintereinander vorkommen, desto ausgeprägter die visuelle Repräsentation. Abbildung 3 zeigt eine manuelle Interpretation der grafischen Elemente aus dem Beispiel nach sieben Generationen. Eine Generation bezieht sich dabei auf die Anzahl der Anwendungen der Produktionen auf die Zeichenkette. Beginnend bei eins = Startwort. Wie zu erkennen ist, existiert ein Interpretationsspielraum. Was bedeutet leicht nach innen? Wie stark ist die Vertiefung ausgeprägt usw. Diesen Spielraum eliminiert die grafische Darstellung mithilfe der Turtle-Interpretation (siehe. V-A).

D. Limitationen

Per Definition verzichten 0L-Systeme auf das festlegen von Terminalen, was zur Folge hat, dass die generierten Zeichenketten von L-Systemen unendlich wachsen können und keine Endzustände in Organismen oder Zellsterben von Grund auf implementieren. Ein derartiges Verhalten implementieren erweiterte L-Systeme, die wiederum Bedingungen benötigen und dadurch Kontextsensitivität erlangen [1], [6]. Auch im Hinblick auf die physikalischen Eigenschaften von Pflanzen hat ein 0L-System keine Antwort. Das 0L-System berücksichtigt nicht, aus welchem Material der Stamm der Pflanze ist oder wie viel Tragkraft der Ast eines Baumes hat. Das erschwert die Verifikation des Modells im Sinne der korrekten Abbildung der realen Welt. Im Endeffekt sind nur Annäherungen möglich.

IV. L-Systeme und Chomsky-Hierarchie

Zur besseren Einschätzung der L-Systeme in der Chomsky-Hierarchie gilt es, eine Behauptung aufzustellen und einen Beweis durch Widerspruch durchzuführen. Dies geschieht mit dem Vergleich der Eigenschaften von 0L-Systemen mit den Eigenschaften der verschiedenen Sprachtypen. Dabei ist die Behauptung generisch nach dem Muster '0L-System ist kein Sprachtyp X'. Somit ist die Annahme, dass 0L-Systeme vom Sprachtyp X sind, zu beweisen. Durch dieses Konstrukt müssen alle Eigenschaften des Sprachtyps X den Eigenschaften der 0L-Systeme entsprechen. Ist dies bei einer Eigenschaft nicht der Fall, entsteht ein Widerspruch und die Annahme ist validiert.

Die für 0L-Systeme zur Beweisführung betrachteten Eigenschaften sind unter Abschnitt III-C1 zu finden.

Im Allgemeinen gilt für jede Reductio ad absurdum, dass sowohl 0L-Systeme und alle Sprachtypen einen Startpunkt und ein Alphabet bestehend aus Symbolen besitzen. Diese Eigenschaften werden nicht weiter geprüft, da der Unterschied in den Sprachtypen in den Einschränkungen und Produktionsregeln liegt.

Die Prüfung der Abgeschlossenheit von L-Systemen unter den Operationen Vereinigung, Stern, Konkatenation, Komplementbildung und Schnitt stellt keine vollständige Beweisführung dar, da diese in Fachliteraturen zu formalen Sprachen bereits durchgeführt wurden und über den eigentlichen Aspekt der Arbeit hinaus gehen. Die Anmerkungen zur Abgeschlossenheit sollen lediglich anmerken, wie es möglich wäre dies umzusetzen.

A. 0L-System ist kein Sprachtyp-1

Nach dem festgelegten Konstrukt ist die Annahme: 0L-System ist vom Sprachtyp-1 der Chomsky-Hierarchie.

Vergleich	**0L-System**	**Typ-1**	$\surd$ $\times$
Prod.Regeln	$A \rightarrow \alpha$	$\alpha A \beta \rightarrow \alpha \gamma \beta$	$\times$
Restriktionen	$A \in V$, $\alpha \in (V \cup \Sigma)^*$	$A \in V$ $\alpha, \beta \in (V \cup \Sigma)^*$ $\gamma \in (V \cup \Sigma)^+$ $\lvert\gamma\rvert \geq 1$	$\times$
Kontext	kontextfrei	Kontextsensitiv	$\times$

Tabelle II
VERGLEICH DER EIGENSCHAFTEN ZWISCHEN 0L-SYSTEMEN UND DEM SPRACHTYP-1

Neben den allgemeinen Übereinstimmungen existieren zwei weitere Einschränkungen. Sowohl bei den 0L-Systemen, als auch den Typ-1 Sprachen gilt $A \in V$ und $\alpha \in (V \cup \Sigma)^*$. Allerdings unterscheiden sich die Produktionsregeln und der Kontext zwischen kontextfreien 0L-Systemen und kontextsensitiven Typ-1 Sprachen (vgl. Tabelle II). Dies widerlegt die Annahme und validiert die Behauptung. Ein 0L-System ist kein Sprachtyp-1 und kann daher auch nicht vom Sprachtyp-0 sein.

B. 0L-System ist kein Sprachtyp-2

Zum Bewies durch Reductio ad absurdum, ist bei dieser Behauptung die Annahme: 0L-System ist vom Sprachtype-2 der Chomsky-Hierarchie.

Vergleich	**0L-System**	**Typ-2**	$\surd$ $\times$
Prod.Regeln	$A \rightarrow \alpha$	$A \rightarrow \gamma$	$\surd$
Restriktionen	$A \in V$, $\alpha \in (V \cup \Sigma)^*$	$A \in V$ $\gamma \in (V \cup \Sigma)^*$	$\surd$
Kontext	kontextfrei	kontextfrei	$\surd$

Tabelle III
VERGLEICH DER EIGENSCHAFTEN ZWISCHEN 0L-SYSTEMEN UND DEM SPRACHTYP-2

Die allgemeinen Eigenschaften sind bereits geprüft und stimmen überein. Wie der Tabelle III zu entnehmen ist, sind die Produktionsregeln, die Restriktionen und der Kontext identisch. Da 0L-Systeme mit den Eigenschaften von Typ-2 Sprachen übereinstimmen, gilt es zu prüfen, ob die Abgeschlossenheit der Systemen unter den Operationen Vereinigung, Konkatenation, Stern sich decken. Bei der Komplementbildung und dem Schnitt sollten 0L-Systeme nicht abgeschlossen sein. Diese Eigenschaften zur Abgeschlossenheit lassen sich aus der allgemeinen Theorie formaler Sprachen und L-Systemen ableiten [3], [6].

Vereinigung $L_1 \cup L_2$: Zwei separate 0L-Systeme erzeugen die Sprachen L_1 und L_2. Durch die Vereinigung beider entsteht eine neue 0L-Grammatik durch einführen eines neuen Startsymbols, welches entweder die Produktionsregeln von L_1 oder L_2 wählt. A_1 und A_2 sind die jeweiligen Automaten der Sprachen L_1 und L_2. Q_1 und Q_2 beschreiben dabei die korrespondierenden Zustände der Automaten.

- Startzustand: $q_0 \xrightarrow{\varepsilon} q_{start1}$ und $q_0 \xrightarrow{\varepsilon} q_{start2}$.
- Übergänge: A_1 und A_2 bleiben unverändert.
- Endzustände: Vereinigte Menge aus A_1 und A_2

Konkatenation L_1L_2: Gegeben sind zwei separate 0L-Systeme L_1 und L_2. Durch Festlegung der Reihenfolge zur Anwendung der Regeln. Kann aus diesen separaten 0L-Systemen eine neue gemeinsame Grammatik entstehen. Diese führt bspw. zuerst die Produktionsregeln von L_1 aus und danach die von L_2. A_1 und A_2 sind die jeweiligen Automaten der Sprachen L_1 und L_2. Q_1 und Q_2 beschreiben dabei die korrespondierenden Zustände der Automaten.

- Startzustand: $q_0 \xrightarrow{\varepsilon} q_{start1}$.
- Übergänge: A_1 und A_2 bleiben unverändert, alle Endzuständen aus A_1 haben einen ε-Übergang zu q_{start2}.
- Endzustände: Alle Endzustände von A_2.

Kleene-Stern L^*: Ein 0L-System erzeugt die Sprache L. Durch 0..* Konkatenationen von Wörtern aus L, entsteht ein erweitertes 0L-System, welches die Wiederholung von Regeln erlaubt. A ist der Automaten der Sprachen L. Q die korrespondierenden Zustände des Automaten.

- Startzustand: q_0, der auch Endzustand ist.
- Übergänge: A bleiben unverändert, alle Endzuständen aus A haben einen ε-Übergang zu q_0.
- Endzustände: q_0 und alle bisherigen Endzustände.

Typ-2 Sprachen der Chomsky-Hierarchie gelten unter der Komplementbildung und dem Schnitt als nicht abgeschlossen.

Schnitt $L_1 \cap L_2$: Laut Definition ist a^nb^n sowohl eine Chomsky-Typ-2-Sprache als auch ein 0L-System, ebenso wie a^nc^n. Definieren diese beiden Sprachen $L_1 = a^nb^n \mid n \geq 0$ und $L_2 = a^nc^n \mid n \geq 0$, ergibt ihr Schnitt die neue Sprache $L_3 = L_1 \cap L_2 = a^nb^nc^n \mid n \geq 0$.

Der Schnitt lässt sich wie folgt begründen: Ein Wort w gehört zu L_3, wenn es in beiden Sprachen L_1 und L_2 enthalten ist.

Aus L_1 folgt, dass w die Struktur a^nb^n haben muss. Aus L_2 folgt, dass w die Struktur a^nc^n haben muss. Da ein Wort gleichzeitig beide Bedingungen erfüllen muss, ergibt sich $w = a^nb^nc^n$. Dies zeigt, dass L_3 die Sprache $a^nb^nc^n \mid n \geq 0$ beschreibt. Wie im Beispiel in Abschnitt III-B3 erläutert, gehört diese Sprache zur kontextsensitiven Klasse (Typ-1) der Chomsky-Hierarchie. Da jedoch 0L-Systeme wie Typ-2-Sprachen im Schnitt nicht abgeschlossen sind, kann ein 0L-System keine Sprache des Sprachtyps 1 darstellen. Dies verdeutlicht die Grenze der Ausdrucksmächtigkeit von 0L-Systemen.

Komplementbildung $\overline{L}$: Für die Komplementbildung werden alle Wörter der Sprache L benötigt, die nicht in der Sprache L enthalten sind. Wird hier dasselbe Beispiel wie im Schnitt untersucht a^nb^n, so sind alle Wörter enthalten, deren Anzahl von a's und b's unterschiedlich ist oder deren Reihenfolge von a's und b's anders ist wie z.B. 'aabab'. 0L-Systeme sind ein spezieller Typ der L-Systeme, die kontextfrei agieren und keine rekursiven oder kontextabhängigen Strukturen erzeugen können. Ihnen fehlt die Flexibilität für komplexe Operationen zur Komplementbildung.

Nachdem die Eigenschaften von 0L-Systemen und Typ-2 Sprachen übereinstimmen und die Abgeschlossenheit unter denselben Operationen vorliegt, ist die Annahme nicht widerlegt. Demzufolge ist die Behauptung falsch und ein 0L-System ist vom Sprachtyp-2 der Chomsky-Hierarchie.

C. 0L-System ist kein Sprachtyp-3

Für die Beweisführung dieser Behauptung ist die Annahme: 0L-System ist vom Sprachtyp-3 der Chomsky-Hierarchie.

Vergleich	**0L-System**	**Typ-3**	$\surd$ $\times$
Prod.Regeln	A $\rightarrow \alpha$	$A \rightarrow aB \vert Ba$ $A \rightarrow a$	$\times$
Restriktionen	A $\in$ V, $\alpha \in$ (V $\cup \Sigma$)*	rechts- od. linkslinear	$\times$
Kontext	kontextfrei	kontextfrei, linear	$\surd$

Tabelle IV
VERGLEICH DER EIGENSCHAFTEN ZWISCHEN 0L-SYSTEMEN UND DEM SPRACHTYP-3

Bei Typ-3 Sprachen zeigt sich, dass diese mit 0L-Systemen teilweise übereinstimmen. Es ist durchaus möglich ein 0L-System so zu definieren, dass es alle Kriterien einer Typ-3 Sprache abbildet. Das Alphabet und die zugehörigen Produktionsregeln sind dementsprechend zu wählen. Dennoch sind 0L-Systeme mächtiger als Typ-3 Sprachen. Dies zeigen die Produktionsregeln und deren Restriktionen (vgl. Tabelle IV). Bei 0L-Systemen können die Produktionsregeln rechts-, links- und nicht linear sein. Ein solches Beispiel ist das 0L-System aus Gleichung 2. Die Produktionsregeln aus diesem Beispiel beinhalten keinerlei Terminale. Das bedeutet: Über die Zeit hinweg ist die Zeichenkette unendlich lang durch ständiges Ersetzen der Variablen und bildet nicht die Form A $\rightarrow$ aB | BA bzw. A $\rightarrow$ a, wobei a $\in$ T. Dadurch ist die Annahme widerlegt und es handelt sich bei 0L-Systeme nicht um einen Sprachtyp-3 der Chomsky-Hierarchie. Ausnahme: Die Produktionsregeln und das Alphabet wird so gewählt, dass die Eigenschaften von Typ-3 zutreffen. Allerdings stellt sich die Frage, wie sinnhaft die erzeugten Zeichenketten sind, in der Regel bilden sich Fraktale (vgl. Abbildung 5) oder Pflanzen, die unendliche Länge in einem endlichen Raum darstellen [1], [2], [6].

$$\begin{aligned} &\text{Axiom: } X \\ &\quad X \rightarrow F[-X][+X]F[X] \\ &\quad F \rightarrow FF \end{aligned} \tag{2}$$

D. Schlussfolgerung zu L-Systeme und Chomsky

Durch die Vergleiche mit den verschiedenen Sprachtypen lässt sich feststellen, dass 0L-Systeme mit den kontextfreien Sprachen des Typ-2 zu vereinbaren sind [1]. Trotz der Parallelen zwischen den Formalismen sind L-Systeme keine direkte Sprachtypen. Sie weisen gleiche Eigenschaften und Verhaltensweisen auf. Sie sind nicht mächtiger als Chomsky-Sprachtypen [6]. Wie Abbildung 4 zeigt, sind 0L-Systeme im Bereich kontextfrei und abwärts anzusiedeln (Mit Ausnahmen).

Die Grenze zwischen kontextfrei-kontextsensitiv können L-Systeme überschreiten, weil sie als spezifische Sprache unter

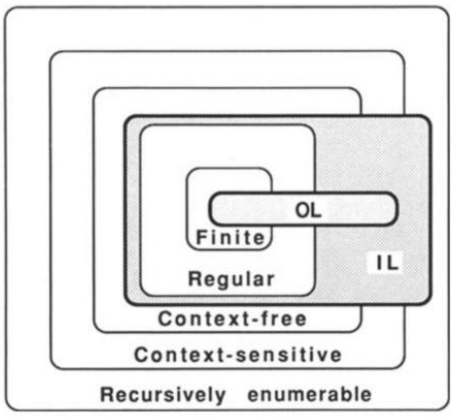


Abbildung 4. Visuelle Darstellung zur Kategorisierung der L-Systeme in der Chomsky-Hierarchie [6]

Betrachtung von Parallelität zur Modellierung von Wachstumsprozessen bei Pflanzen in der computergestützten Botanik entworfen wurden [6]. Hingegen stellen Chomsky-Sprachtypen eine hierarchischen Klassifizierung formaler Sprachen und Grammatiken dar [8]. Sie wenden die Produktionsregeln sequentiell an [7], [8], [10]. L-System dagegen wenden ihre Produktionsregeln parallel an [1], [6], [11]. Diese Unterschiede erschweren eine eindeutige 'Klassifizierung' der L-Systeme in der Chomsky-Hierarchie.

Ein weiter Unterschied zur klassischen Beschreibung einer formalen Sprache mit einer formalen Grammatik ist das Vorhandensein von Terminalen. So lange nicht explizit angegeben und in das L-System integriert, existieren in einem L-System keine Terminale. Terminale können in ein L-System implementiert werden und suggerieren dabei Zellsterben bzw. den Endzustand eines Organismus an dieser Stelle (Blüte einer Pflanze). Eine detailliertere Beschreibung zur Verwendung von Terminalen zur Simulation von Zellsterben oder Modellierung von Endzuständen würde weitere theoretische Grundlagen von parametrisierten, partiellen oder $\langle m, n \rangle$L-Systemen voraussetzen. Dies ist nicht Teil dieser Arbeit, da es den Umfang deutlich überschreiten würde. Nähere Informationen dazu finden sich in P. Prusinkiewicz und A. Lindenmayers Werk 'The algorithmic beauty of plants' [6].

Durch ihren spezifischen Entwicklungskontext kann der mathematische Formalismus L-System als Interpreter einer Pflanze angesehen werden. Durch die Verbindung des Formalismus mit einer grafischen Interpretation sind L-Systeme in speziellen Anwendungsbereichen mächtiger als bspw. reine Typ-2 Sprachen. Allerdings ist der direkte Vergleich schwierig, da Typ-2 Sprachen nicht mit einer grafischen Interpretation in Verbindung gebracht wurden. Im Vergleich der Anwendungsbereiche schneiden Typ-2 Sprachen besser ab, da diese ein breites Spektrum der Anwendung bieten und A. Lindenmayer seine L-Systeme ausschließlich für den Anwendungsfall des Pflanzenwachstums und computergenerierten Pflanzen entwickelt hat. Die Kopplung zur grafischen Interpretation macht aus ihnen ein spezifisches und hilfreiches Werkzeug in diesem Anwendungsgebiet.

V. Anwendungsgebiet Computergrafik

Mit der formalen Definition von L-Systemen fällt eine grafische Interpretation des generierten Strings schwer, da diese für eine grafische Darstellung als Variable mit fester Zuordnung anzusehen sind. Um diesen Umstand zu verbessern, verwendet P. Prusinkiewicz und A. Lindenmayer die Turtle Interpretation [6].

A. Turtle Interpretation

Die beschriebene Repräsentation im Beispiel zu 0L-Systemen III-C1 ist möglich, aber erfüllt nicht alle Bedürfnisse zur visuellen Darstellung. Es ist nicht genau festgelegt, ob diese Repräsentation für a,b,c,d,k immer zutreffend sind. Um andere Modelle dazustellen etablierte sich die Verwendung der Turtle Interpretation [12].

Der Zustand einer Turtle ist definiert durch das Tripel (x,y,α), wobei x,y kartesische Koordinaten repräsentieren. Die Position der Turtle und der Winkel α wird 'heading' genannt und ist interpretiert als Richtung in welche sich die Turtle zeigt. Mit der Schrittweite d und dem Winkel-Inkrement δ kann die Turtle auf Anweisungen wie folgt reagieren [2], [6], [11]:

- F : Bewegung nach vorne der Länge d. Der Zustand der Turtle ändert sich zu (x',y',α), mit $x' = x + d \cdot \cos \cdot \alpha$ und $y' = y + d \cdot \sin \cdot \alpha$. Ein Linien Segment zwischen (x,y) und (x',y') wird gezeichnet.
- f Bewegung nach vorne um die Länge d ohne eine Linie zu zeichnen.
- + Drehung nach links um den Winkel δ. Der nächste Zustand ist (x,y, $\alpha + \delta$). Die positive Orientierung der Winkel ist gegen den Uhrzeigersinn.
- - Drehung nach rechts um den Winkel δ.

Mit diesen Festlegungen der Bedeutung der grafischen Interpretation ist es möglich Strukturen zu zeichnen, allerdings ähneln diese noch keiner Pflanze (vgl. Abbildung 5).

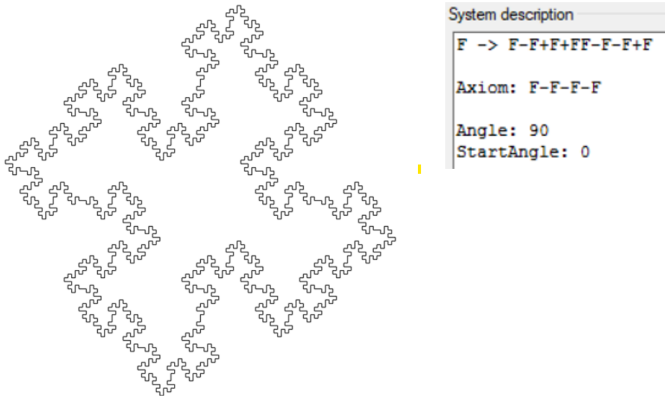


Abbildung 5. Generierung einer quadratischen Koch Insel mit Generation n = 3 [2], [6]

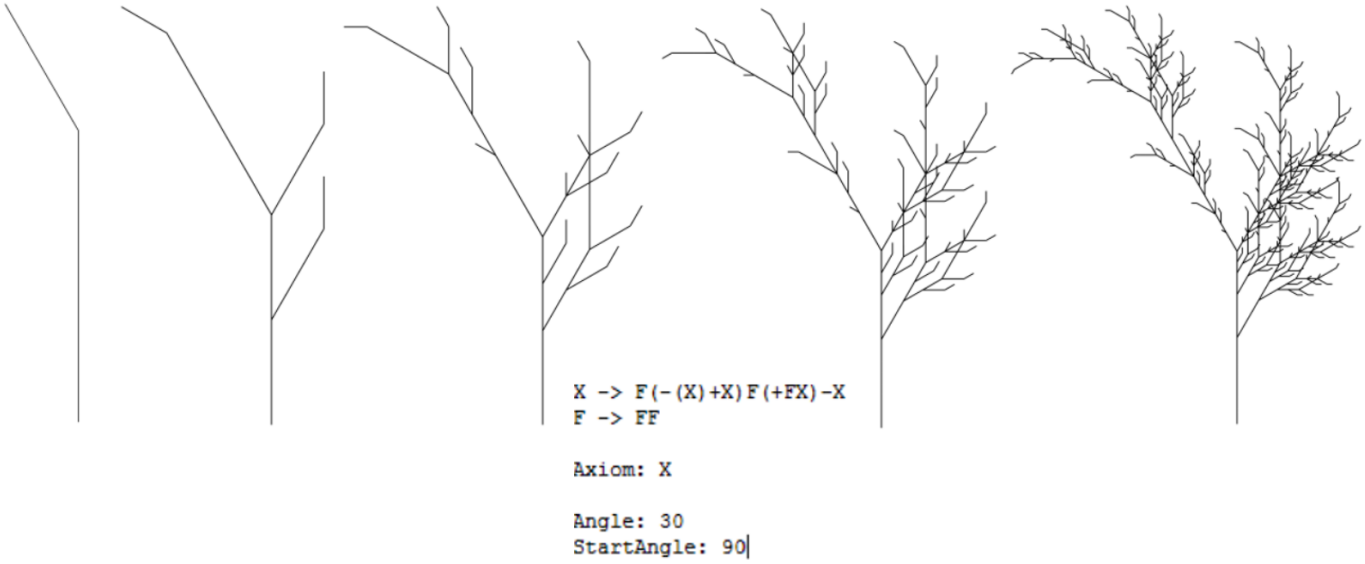
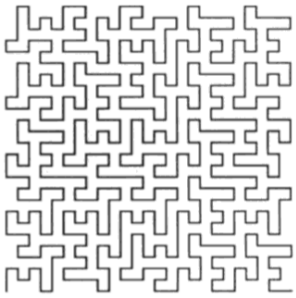


Abbildung 6. Wachstumsprozess eines Krauts über fünf Generationen



b
n=2, δ=90°
$-F_r$
$F_l \rightarrow F_lF_l-F_r-F_r+F_l+F_l-F_r-F_rF_l+$
$F_r+F_lF_lF_r-F_l+F_r+F_lF_l+$
$F_r-F_lF_r-F_r-F_l+F_l+F_rF_r-$
$F_r \rightarrow +F_lF_l-F_r-F_r+F_l+F_lF_r+F_l-$
$F_rF_r-F_l-F_r+F_lF_rF_r-F_l-$
$F_rF_l+F_l+F_r-F_r-F_l+F_l+F_rF_r$

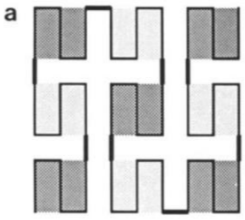
Abbildung 7. Quadratische Gosper Kurve [6]

1) Erweiterung: Kantenumschreibung: Die Interpretation wird erweitert um F_l und F_r. Die entstehende Kante deutet am Ende eine Links- oder Rechtsdrehung an [6]. Diese Erweiterung zeigt in Abbildung 7 'weichere' Ecken mit Rundungen, aber ist nicht ausreichend für eine Simulation eines Wachstumsprozesses einer Pflanze.

2) Erweiterung: Knotenumschreibung: Das System wird um L_n und R_n erweitert. Diese beschreiben Knotenpunkte im System mit einem eigenen Regelset zur visuellen Darstellung des Knotenpunkts und der Verbindung zum restlichen Graphen [6]. Wie Abbildung 8 zeigt, sind nun Unterteilungen mit Knotenpunkten möglich. Auch diese Erweiterung ist noch nicht ausreichend.

3) Erweiterung: Verzweigung: Für eine bessere Abbildung auf den Anwendungsfall der computergestützten Botanik und Pflanzenwachstum zu ermöglichen, braucht es Mechanismen zur Verzweigung. Dies ist z.B. bei Bäumen ein natürlicher Bestandteil des Wachstumsprozesses. Mit den bisherigen Erweiterungen der grafischen Interpretation können zwar Überschneidungen und Abzweigungen realisiert werden, aber hier muss die Turtle immer wieder den Weg zurückfinden und dort weiter zeichnen. Deshalb wurden eckige Klammern eingeführt und in Verbindung mit einem pushdown Stack kann die Turtle den aktuelle Zustand im Stack speichern. Die Informationen im Stack können je nach Implementierung abweichen. Neben den Koordinaten zum Zustand kann der Stack z. B. Daten zur Farbe und dicke der Linie speichern. Die Turtle schreibt durch push Operation den Zustand in den Stack [6], [11].

- '[' (geöffnete eckige Klammer) Schreibe den aktuellen Zustand der Turtle in einen pushdown Stack.
- ']' (geschlossene eckige Klammer) Lese den Zustand des letzten Eintrags im Stack und setze den gelesenen



a n=2, δ=90°
L
L→LFRFL-F-RFLFR+F+LFRFL
R→RFLFR+F+LFRFL-F-RFLFR

Abbildung 8. FASS Kuve mit 3x3 Kacheln [6]

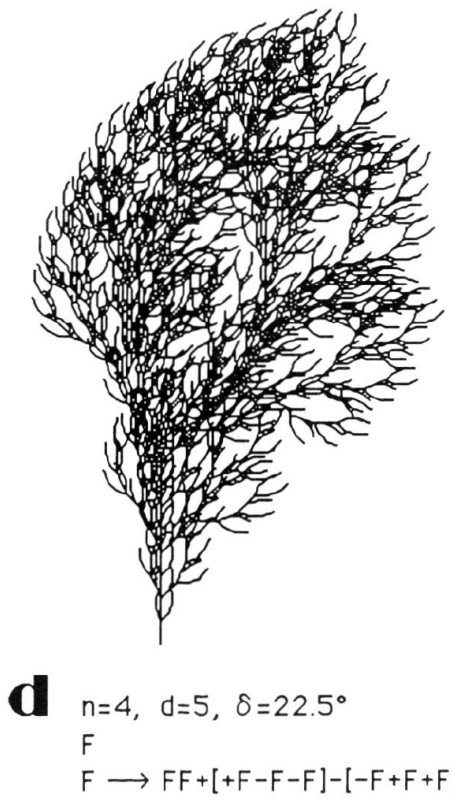


Abbildung 9. Generierter Busch durch ein 0L-System unter der Verwendung von Verzweigung [2]

Zustand als aktuellen Zustand der Turtle. Es wird keine Linie gezeichnet.

Abbildung 9 zeigt, dass die Verzweigung es erlaubt pflanzenähnliche Strukturen zu generieren. Diese Erweiterung, in Kombination mit den bisherigen, ermöglicht es den L-Systemen ihr Potential in grafischer Interpretation zu zeigen. In der Abbildung 6 ist der Wachstumsprozess eines Krautes in fünf Generationen zu sehen. Die verwendete Software [13] zur Generierung dieser Grafik entwickelte Prof. Dr. E. Weitz und verwendet statt eckiger Klammern runde Klammern in der Verzweigung.

4) Erweiterung: Farben: Wie bisher gesehen, können kleine Veränderungen im Vokabular große Auswirkungen auf das Endergebnis haben. So ist es möglich mit bspw. Kleinbuchstaben (n für braun und g für grün) eine Farbcodierung in das L-System zu integrieren. Durch diese Erweiterung kann ein L-System (vgl. Abbildung 10) einen Busch mit grünen Blättern wie in Abbildung 11 erzeugen [11].

5) Erweiterung: Dreidimensionaler Raum: Die aktuelle Ausrichtung der Turtel basiert auf drei Vektoren $\vec{L}$,$\vec{H}$ und $\vec{U}$. Die Vektoren haben eine Einheitslänge und sind senkrecht zueinander. Beschrieben durch die 3x3 Rotationsmatrizen [6]:

$$R_U(\alpha) = \begin{bmatrix} \cos\cdot\alpha & \sin\cdot\alpha & 0 \\ -\sin\cdot\alpha & \cos\cdot\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3)$$

Matrix 3 beschreibt die Rotation um die z-Achse im dreidimensionalen Raum um den Winkel α.

$$R_L(\alpha) = \begin{bmatrix} \cos\cdot\alpha & 0 & -\sin\cdot\alpha \\ 0 & 1 & 0 \\ \sin\cdot\alpha & 0 & \cos\cdot\alpha \end{bmatrix} \quad (4)$$

Matrix 4 beschreibt die Rotation um die y-Achse im dreidimensionalen Raum.

$$R_H(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\cdot\alpha & -\sin\cdot\alpha \\ 0 & \sin\cdot\alpha & \cos\cdot\alpha \end{bmatrix} \quad (5)$$

Matrix 5 beschreibt die Rotation um die x-Achse im dreidimensionalen Raum.

Zur Orientierung der Turtle im Raum gelten folgende Symbole [6]:

- +: Linksdrehung um den Winkel δ unter der Verwendung der Rotationsmatrix $R_U(\delta)$

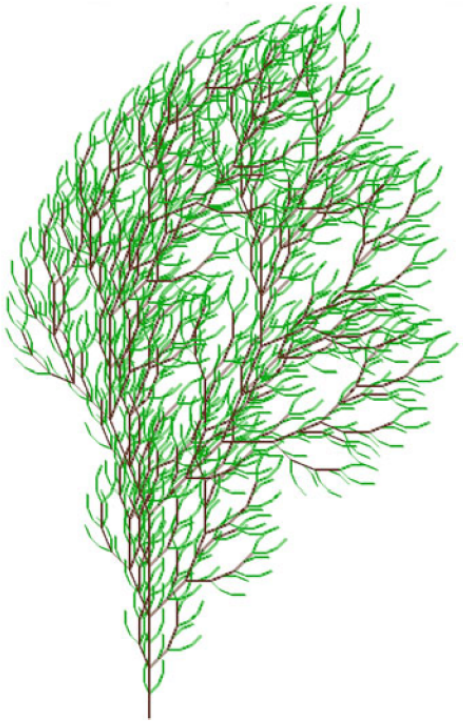


Abbildung 10. Ein Busch mit Blättern und Einfärbung [11]

```
Axiom: gF
StartAngle: 90
Angle: 22.5

Regeln:
F -> nFF-g(-F+F+F)+g(+F-F-F)
```

Abbildung 11. L-System-Regeln für einen Busch mit grünen 'Blättern' [11]

Abbildung 13. 3D Modell einer Blume mit dem 0L-System aus 12 [6]

n=5, δ=18°

ω : **plant**
p_1 : **plant** → **internode** + [**plant** + **flower**] − − //
[− − **leaf**] **internode** [+ + **leaf**] −
[**plant flower**] + + **plant flower**
p_2 : **internode** → F **seg** [// & & **leaf**] [// ∧ ∧ **leaf**] F **seg**
p_3 : **seg** → **seg** F **seg**
p_4 : **leaf** → [' { +f−ff−f+ | +f−ff−f }]
p_5 : **flower** → [& & & **pedicel** ‘ / **wedge** //// **wedge** ////
wedge //// **wedge** //// **wedge**]
p_6 : **pedicel** → FF
p_7 : **wedge** → [‘ ∧ F] [{ & & & & −f+f | −f+f }]

Abbildung 12. 0L-System einer 3D modellierten Blume [6]

- -: Rechtsdrehung um den Winkel δ unter der Verwendung der Rotationsmatrix $R_U(-\delta)$
- &: Höhe nach unten um den Winkel δ verschieben, unter der Verwendung der Rotationsmatrix $R_L(\delta)$
- $\wedge$: Höhe nach oben um den Winkel δ verschieben, unter der Verwendung der Rotationsmatrix $R_L(-\delta)$
- \: Bewegung nach links um den Winkel δ unter der Verwendung der Rotationsmatrix $R_H(\delta)$
- /: Bewegung nach rechts um den Winkel δ unter der Verwendung der Rotationsmatrix $R_H(-\delta)$
- |: Drehung um 180° unter der Verwendung der Rotationsmatrix $R_U(180°)$

Die Abbildung 12 zeigt ein beschreibendes 0L-System einer Blume. In diesem Beispiel wird mit Variablen plant, internode, flower, seg, leaf, predicel und wedge gearbeitet. Jede dieser Variablen erhält eigene Produktionsregel, um Überblick im L-System beizubehalten. Es zeigt auch die Anwendungen der Erweiterungen für den dreidimensionalen Raum und generiert die Blume aus Abbildung 13.

Welches grafische Element die Turtle auf dem Weg zeichnet bleibt der Implementierung überlassen. Diese Eigenheit erlaubt eine flexible Darstellung. Die eingeführten Symbole bilden die Navigation der Turtle durch den Ruam (2D oder 3D). Dabei kann wie in Abbildung 13 ein 'leaf' grafisch länglich und spitz zulaufend oder gestaucht und rund sein. Diese Interpretation obliegt dem Anwender. Dadurch lassen sich mit einem kleinen Alphabet Σ und wenigen Produktionsregeln bereits realistische Pflanzen beschreiben und visuell darstellen.

B. Modellierung realitätsnaher Pflanzen

In der Funktionsweise unterschiedlich, aber in der grafischen Interpretation ähnlich, erlauben partielle L-Systeme realitätsnahe grafische Abbildungen einer Pflanze (vgl. Abbildung 14). Deren Theorie und Funktionsweise ist nicht Teil dieser Arbeit, dennoch ist deren Mächtigkeit und das grafische Abbild eines solchen L-Systems erwähnenswert. Dazu gehören weitere Theorien zu Wachstumsfunktionen, Terminierungen (Zellsterben oder Endstadien) und weitere Diese beschreiben P. Prusinkiewicz und A. Lindenmayer im Buch 'The algorithmic beauty of plants' aus dem Jahre 1990 [6].

VI. Diskussion

Die bisher erläuterte Mächtigkeit von L-Systemen wird durch deren Limitationen geschmälert. Es ist zwar möglich realistische Modelle für Pflanzen zu erstellen, aber dies kann schwierig und zeitintensiv sein. Die Regeln für die korrekte

Abbildung 14. 3D-Modell eines Apfelbaumzweigs [6]

Erstellung der Verzweigungen, Blattbildungen usw. soll mit echten Pflanzen übereinstimmen. Hierfür braucht es tiefe Einblicke in die DNA und den Wachstumsprozess der zu modellierenden Pflanze. In der Literatur [1], [2], [6], [11] finden sich zwar Beispiele für Pflanzenmodelle, allerdings sind diese auf den eigenen Anwendungsfall anzupassen. Des Weiteren erfordern realitätsnahe Modelle physikalische Simulationen des erstellten Konstrukts. L-Systeme bieten keine Möglichkeit zur physikalischen Verifikation oder Simulation des Modells. Darüber hinaus sind sie kein universeller Standard und leben in einem Feld mit Konkurrenztechnologien wie z. B. andere prozedurale Modellierungswerkzeuge (Houdini [14], SpeedTree [15] oder Physiksimulationen). Es erfordert meist eine Kombination aus mehreren Verfahren.

Auf der anderen Seite ist durch ihre Leichtfüßigkeit eine schnelle Implementierung möglich. In der prozeduralen Generierung kann ein L-System mit wenigen Regeln bereits komplexe Strukturen erzeugen. Durch kleine Anpassungen der Startbedingungen oder in den Produktionsregeln besitzen sie eine große Flexibilität. Das Modell ist an verschiedene Pflanzenarten oder Wachstumsmodelle anpassbar. Zudem ist der Speicheraufwand und die Rechenleistung, je nach Aufgabenstellung, gering. In einfachen Beispielen, wie in dieser Arbeit gezeigt, reicht es aus Regeln und Startbedingungen zu speichern. Trotzdem gilt: Je größer die Zahl der Generation und grafischen Elementen bzw. Interpretationsmöglichkeiten, desto größer die benötigte Rechenleistung.

VII. Fazit

L-Systeme lassen sich aufgrund ihrer Eigenschaften mit Sprachtypen zwar Vergleichen, aber nicht direkt zu einem Sprachtyp hinzufügen. Diese Beobachtung resultiert aus dem Vergleich der 0L-Systeme mit Typ-2 Sprachen der Chomsky-Hierarchie. Deren größter Unterschied in der Anwendung der Produktionsregeln liegt. L-Systeme wenden ihre Regeln parallel und Chomsky Sprachtypen wenden ihre sequenziell an. Auch ihr spezieller Anwendungsfall erschwert es die L-Systeme in einen konkreten Sprachtypen einzuordnen, da L-Systeme für die computergestützte Botanik zur Simulation von Wachstumsprozessen entwickelt wurden. Trotz all dem ist es ein mathematischer Formalismus und baut auf den Grundlagen formaler Sprachen und Grammatiken auf. A. Lindenmayer ist nur einen Schritt weiter gegangen und verband dies mit den wissenschaftlichen Disziplinen Biologie und Informatik. Für den Anwendungsbereich Computergrafik eignen sich L-Systeme sehr gut. Sie liefern vorgefertigte Regeln zur Ableitung eines Systems über ein Axiom und bringen gleichzeitig die Definitionen zur grafischen Interpretation mit. Dabei lassen L-Systeme genügen Konfigurationsspielraum, um eigene Gedanken mit einfließen zu lassen. Z. B. Farbgebung und Linieninterpretation. Auch wenn L-Systeme sich nicht als Industrie-Standard etabliert haben, sind sie ein spezialisiertes Werkzeug zur Generierung von Pflanzen, natürlichen Strukturen und Fraktalen. Z. B. könnten sie durch ihre leichte Verwendung und Flexibilität dazu verwendet werden, Arten von Vegetationen in Filmen oder Spielen zu kreieren. Die tatsächliche Implementierung der Vegetationen sollte dennoch SpeedTree [15], einem weitverbreitetem Tool in der Industrie, überlassen werden.

Des Weiteren ist die Klassifizierung von L-Systemen als formaler Sprachtyp eine Grundlage für mathematische Beweisführung, da Rückschlüsse aus der Chomsky-Hierarchie gezogen werden können. Der größere Vorteil ist allerdings die bessere Modellierung von Pflanzenwachstumsprozessen. Durch die formale Klassifizierung entsteht die Grundlage für einen Compilerbau im Bereich der computergestützten Botanik und der Computergrafik. Die Zuordnung kann dabei helfen effiziente Algorithmen zur Echtzeitsimulation zu entwickeln.

Neben dem Klassiker 'The algorithmic beauty of plants' [6] finden sich in erster Linie keine aktuellen wissenschaftlichen Arbeiten. Dennoch ist [6] das Standardwerk für grundlegende Theorien und Modelle zu diesem Thema. Nach längerer Recherche stelle sich heraus, dass L-Systeme im wissenschaftlichem Bereich verbreitet [16] sind.

Literatur

[1] A. Lindenmayer, "Developmental algorithms for multicellular organisms: A survey of l-systems," *Journal of Theoretical Biology*, vol. 54, no. 1, pp. 3–22, 1975, verfügbar unter: https://www.sciencedirect.com/science/article/pii/S0022519375800518.

[2] P. Prusinkiewicz, "Graphical applications of l-systems," in *Proceedings of graphics interface*, vol. 86, no. 86, 1986, pp. 247–253.

[3] J. E. Hopcroft and J. D. Ullman, *Formal languages and their relation to automata*. USA: Addison-Wesley Longman Publishing Co., Inc., 1969, verfügbar unter: https://dl.acm.org/doi/pdf/10.5555/1096945.

[4] H.-P. Gumm and M. Sommer, *Band 3 Formale Sprachen, Compilerbau, Berechenbarkeit und Komplexität*. Berlin, Boston: De Gruyter Oldenbourg, 2019, abgerufen am: 03.01.2025, https://doi.org/10.1515/9783110442397.

[5] I. f. T. I. Universität Siegen, "Compilerbau – vorlesungsskript," 2022, abgerufen am 03.01.2025, verfügbar unter: https://www.eti.uni-siegen.de/ti/lehre/sommer_2022/compilerbau/cb.pdf.

[6] P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants*. New York: Springer-Verlag, 1990, verfügbar unter: https://link.springer.com/book/10.1007/978-1-4613-8476-2.

[7] L. M. Institut für Informatik, "Grammatiken, probleme und eigenschaften – vorlesungsunterlagen," 2023, abgerufen am 29.12.2024, verfügbar unter: https://www.tcs.ifi.lmu.de/lehre/ss-2023/fsk_de/04-grammatiken-probleme-eigenschaften.pdf.

[8] N. Allott, T. Lohndal, and G. Rey, Eds., *A Companion to Chomsky*. Hoboken, NJ: John Wiley & Sons, 2021, verfügbar unter: https://onlinelibrary.wiley.com/doi/book/10.1002/9781119598732.

[9] L. M. Institut für Informatik, "Kontextsensitive sprachen (typ 1)," 2024, abgerufen am: 01.01.2025, verfügbar unter: https://www.pms.ifi.lmu.de/NFModule/TheoretischeInformatik/Typ01.pdf.

[10] R.-U. Bochum, "Die chomsky-hierarchie," abgerufen am 29.12.2024, verfügbar unter: https://www.ruhr-uni-bochum.de/lmi/lehre/materialien/ti/vorlesung/kap1-1.pdf.

[11] C. Maurer, *Objektbasierte Programmierung mit Go*, 2nd ed. Wiesbaden: Springer Vieweg Wiesbaden, 2023, verfügbar unter: https://link.springer.com/book/10.1007/978-3-658-42014-7.

[12] R. Goldman, S. Schaefer, and T. Ju, "Turtle geometry in computer graphics and computer aided design," 2010, abgerufen am 04.01.2025, verfügbar unter: https://www.cs.wustl.edu/taoju/research/TurtlesforCADRevised.pdf.

[13] T. Weitz, "Aristid lindenmayer und die lindenmayer-systeme," 2024, abgerufen am 10.12.2024, verfügbar unter: https://weitz.de/lindenmayer/.

[14] SideFX, "Houdini official website," 2025, abgerufen am 03.01.2025, verfügbar unter: https://www.sidefx.com/products/houdini/. [Online]. Available: https://www.sidefx.com/products/houdini/

[15] SpeedTree, "Speedtree official website," 2025, abgerufen am 03.01.2025, verfügbar unter: https://store.speedtree.com/. [Online]. Available: https://store.speedtree.com/

[16] A. B. Group, "Algorithmic botany: Papers," 2025, abgerufen am 02.01.2025, verfügbar unter https://algorithmicbotany.org/papers/.