Lindenmayer Systeme*

Markus Holzer

Universität Tübingen, Wilhelm-Schickard-Institut für Informatik, Sand 13
D-72076 Tübingen, Germany

DRAFT

Inhaltsverzeichnis

1 Geschichtlicher Überblick **2**

2 Schreibweisen **4**

3 Kontextfreie L-Systeme **5**

3.1 Grundlagen zu 0L-Systemen 5

3.2 L-Systeme mit Hilfssymbolen 10

3.3 Abschlußeigenschaften von L-Systeme mit Hilfssymbolen 12

3.4 Zur Rolle von λ-Produktionen in L-Systemen mit Hilfssymbolen 16

3.5 L-Systeme mit Hilfssymbolen und die Chomsky Hierarchie 19

3.6 Zum festen Wortproblem von L-Systemen mit Hilfssymbolen . . 22

3.7 Automatenmodelle für L-Systeme mit Hilfssymbolen 29

3.8 Ein anderer Spracherzeugungsmechanismus 32

4 Kontextsensitive Lindenmayersysteme **36**

4.1 Grundlagen zu IL-Systemen 36

4.2 L-Systeme mit Hilfssymbolen und die Chomsky Hierarchie 40

5 Danksagung und abschließende Worte **41**

Literatur **42**

*Ergänzender Inhalt zur Vorlesung „Formale Sprachen“ gehalten im Sommersemster 1997 an der Universität Tübingen, Wilhelm-Schickard-Institut für Informatik.

1 Geschichtlicher Überblick

Datum: 24. April

Irgendwann im Jahre 1968 schlenderte der Biologe Aristid Lindenmayer (1925–1989) den Gang einer Universität entlang, als er aus einem Hörsaal das Wort „algae“ vernahm. Angesichts der Vertrautheit dieses Klanges blieb er stehen und lauschte. Doch es stellte sich heraus, daß der Dozent gar nicht von Algen sprach. Vielmehr handelte es sich um eine Vorlesung über formale Sprachen. „$L(G)$“ hieß das Zauberwort — die Aussprache von „$L(G)$“ und von „algae“ ist im englischen Sprachgebrauch tatsächlich sehr ähnlich. Und so kam es, daß sich Lindenmayer von nun an den Kopf zerbrach, wie die beiden Begriffe unter einen Hut zu bringen seien.

Noch im Jahre 1968 veröffentlichte Lindenmayer zwei Arbeiten unter dem Titel „Mathematical Models for Cellular Interactions in Developments“ [6, 7]. In einem auf sequentiellen Maschinen basierenden Formalismus gelang es ihm, Zellwachstum für verschiedene Organismen zu modellieren. Das berühmteste Beispiel aus diesen Arbeiten ist das Wachstum der Rotalge, *Callithamnion roseum Harvey*, das nachfolgend dargestellt ist.

Vereinfachend wird angenommen, daß sich bei Callithamnion roseum Harvey folgende Forderungen für das Wachstumsverhalten ergeben:

1. Aus den untersten drei oder vier Zellen des „Stammes“ wachsen keine „Äste“.
2. Jede oberhalb liegende Zelle des „Stammes“ ist mit einem „Ast“ besetzt.
3. Unterhalb einer jeden Spitze befinden sich drei oder vier Zellen, die keine Verästelungen aufweisen.
4. Jeder „Ast“ erster oder höherer Ordnung repliziert das Muster des „Stammes“.

Das von Lindenmayer angegebene Wachstumsmodell stellt Zellen durch die Zahlen 1 bis 9 dar und das Wachstum durch folgende Tabelle:

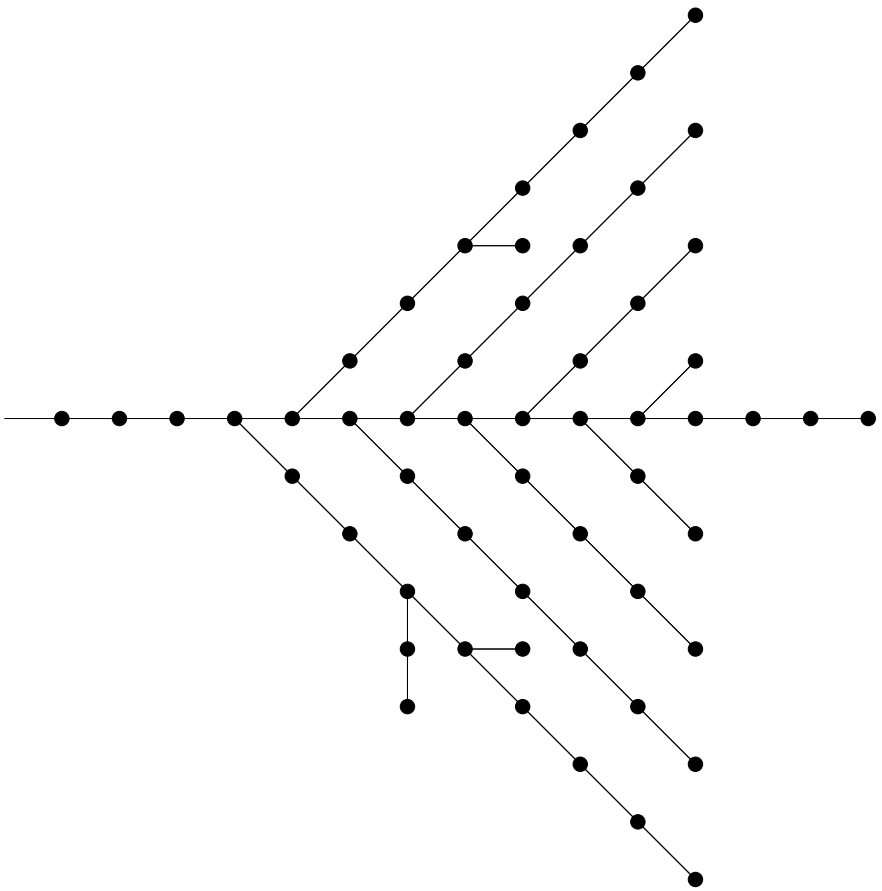
aktueller Zustand	1	2	3	4	5	6	7	8	9
nächster Zustand	2*3	2	2*4	2*5	6*5	7	8	9*[3]	9

Beispielhaft zeigt sich, wie aus der Zelle 1 durch ersetzen *aller* aktuellen Zustände durch deren Nachfolgerzustände folgendes „Zellwachstum“ entsteht:

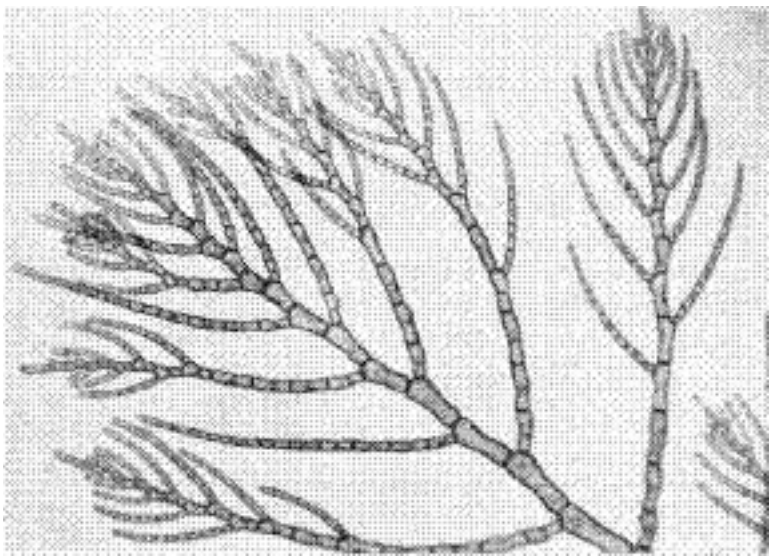
1
23
224
2225
22265
222765
2228765
2229[3]8765

2229[24]9[3]8765
2229[225]9[24]9[3]8765
2229[2265]9[225]9[24]9[3]8765
2229[22765]9[2265]9[225]9[24]9[3]8765
2229[228765]9[22765]9[2265]9[225]9[24]9[3]8765
2229[229[3]8765]9[228765]9[22765]9[2265]9[225]9[24]9[3]8765
2229[229[24]9[3]8765]9[229[3]8765]9[228765]9[22765]9[2265]9[225]9[24]9[3]8765

Interpretiert man die Buchstabe als Linien und die in Klammern gegebenen Worte als Abzweigungen und zeichnet diese abwechselnd nach links oder rechts, dann erhält man für die letzte Zeile folgendes Bild:



Mit ein bißchen Phantasie kommt das dem Original aus der Natur ziemlich nahe:



Nun zurück zur Historie der *Lindenmayer-Systeme*(*L-Systeme*).

Es dauerte nicht lange, und Mathematiker und Informatiker nahmen sich dieses Formalismus an und führten ihn in den 70'er Jahren zu seiner vollen Blüte. Es gab damals kaum ein Kongreß, der nicht von den „Jüngern des L" heimgesucht worden wäre.

Ende der 70'er und Anfang der 80'er kam es dann unaufhaltsam zum Niedergang der Bewegung. Als Lindenmayer 1989 verstarb, war ein Großteil der Informatiker schon zur Tagesordnung zurückgekehrt.

Was bleibt, sind Konzepte, die in Teilbereichen der Bildverarbeitung immer noch ihren Dienst tun, Berge von Schriften über Anwendungen von L-Systemen auf die Theorie der formalen Sprachen, und die Erkenntnis, daß Interdisziplinarität ab und zu doch ganz gehörige Wellen schlagen kann.

Einige wichtige Aspekte der Theorie der L-Systeme wollen wir in diesem Skript darstellen. Zur Ergänzung verweisen wir auf die Bücher von Herman und Rozenberg [3], sowie Rozenberg und Salomaa [9]. OK ←

2 Schreibweisen

Im allgemeinen seien Σ und Δ endliche Alphabete. Das frei erzeugte Monoid über Σ bezeichnen wir mit Σ^* und das leere Wort mit λ. Die Länge eines Wortes w bezeichnen wir mit $|w|$. Offensichtlich gilt $|\lambda| = 0$.

Zwei Sprachen $L_1, L_2 \subseteq \Sigma^*$ bezeichnen wir als gleich, wenn sie modulo dem leeren Wort übereinstimmen, d.h. es muß $L_1 \setminus \{\lambda\} = L_2 \setminus \{\lambda\}$ gelten. Die Gleichheit (modulo dem leeren Wort) wird dann auf Sprachklassen verallgemeinert. Diese Vorgehensweise scheint etwas ungewöhlich, stellt aber die üblichen Weg bei den

Formalen Sprachen dar. Dies kommt daher, da in vielen Fällen λ-Produktionen die generative Mächtigkeit von Grammatiken nicht erhöhen. Ein Beispiel dieser Art stellen reguläre und kontextfreie Chomsky-Grammatiken dar. Aus der Grundvorlseung sollte bekannt sein, daß man zu jeder regulären (kontextfreien) Grammatik G eine Grammatik G' vom selben Typ, jedoch ohne λ-Produktionen, konstruieren kann für die $L(G') = L(G) \setminus \{\lambda\}$ gilt.

Ab und zu werden uns die Klassen der Chmosky Hierarchie begegnen. Diese sind die Familie der regulären, der kontextfreien, der kontextsensitiven und der rekursiv aufzählbaren Sprachen. Wir bezeichnen diese Klassen mit REG, CFL, CS und RE. Aus der Grundvorlseung sollte noch bekannt sein, daß

$$\text{REG} \subset \text{CFL} \subset \text{CS} \subset \text{RE}$$

gilt, d.h. die Klassen bilden eine *echte* Hierarchie. OK ←

3 Kontextfreie L-Systeme

3.1 Grundlagen zu 0L-Systemen

Datum: 24. April
Datum: 6. Mai

Durch das Beispiel der Callithamnion roseum Harvey haben wir schon einige Grundbegriffe für Lindenmayer Systeme kennengelernt. So ist z.B. durch $\{1, \ldots, 9, [,]\}$ das *Alphabet* des Systems gegeben, die Entwicklungsregeln bestimmen die *Produktionen*. Buchstabe 1 ist das *Axiom* mit dem die Entwicklung ihren Anfang nimmt. Da keine Wechselwirkung zwischen Zellen besteht, wird das System *kontextfreies L-System* oder *0L-System* genannt. Das Sytem ist *deterministisch*, da keine Wahl bei den Anwendungen der Produktionen besteht. Außerdem ist es *propagierend* (*fortpflanzend*), da keine Zelle gelöscht wird.

Definiton 3.1 *Ein* 0L-System *ist ein 3-Tupel* $G = (\Sigma, \sigma, \omega)$*, wobei* Σ *ein endliches Alphabet,* $\sigma : \Sigma \to 2^{\Sigma^*}$ *eine endliche Substitution, und* $\omega \in \Sigma^*$ *das Axiom ist.*

Anstelle von $w \in \sigma(a)$ schreiben wir auch $a \to_\sigma w$, wobei—falls keine Mißverständnisse auftreten—σ auch weggelassen wird.

Definiton 3.2 *Es sei* $G = (\Sigma, \sigma, \omega)$ *ein 0L-System,* $u \in \Sigma^*$ *und* $i \in \mathbb{N}_0$*. Wir definieren:*

$$L^i(G, u) = \begin{cases} \{u\} & i = 0 \\ \{\, v \in \Sigma^* \mid \exists x \in L^{i-1}(G, u) : v \in \sigma(x) \,\} & \textit{sonst.} \end{cases}$$

Die von G *erzeugte Sprache wird mit* $L(G)$ *bezeichnet und ist definiert als*

$$L(G) = \bigcup_{i=0}^{\infty} L^i(G, \omega).$$

Um die eingeführte Begriffe etwas besser zu verstehen, zwei kleine Beispiele:

Beispiel 1 Das 0L-System $G = (\{a\}, \sigma, a^2)$ mit $\sigma(a) = \{a, a^2\}$ erzeugt die Sprache $L(G) = \{a^n \mid n \geq 2\}$.

Beispiel 2 Das 0L-System $G = (\{a, b\}, \sigma, ab)$ mit $\sigma(a) = \{abab\}$ und $\sigma(b) = \{\lambda\}$ erzeugt die Sprache $L(G) = \{(ab)^{2^n} \mid n \geq 0\}$. Wir erhalten also mit kontextfreien Produktionen eine nicht-kontextfreie Sprache.

Eine einfache Erweiterung von 0L-Systemen sind 0L-System mit Tafeln. Die Motivation solche Systeme zu definieren kommt wiederum aus der Biologie — die Vorstellung dabei ist, daß bei Pflanzen das Wachstum von der Umgebung, also z.B. vom Tag-Nachtrythmus abhängt.

Definiton 3.3 *Ein* T0L-System *ist ein 3-Tupel* $G = (\Sigma, H, \omega)$*, wobei* Σ *ein endliches Alphabet,* $H = \{\sigma_1, \ldots, \sigma_n\}$ *eine endliche Menge von endlichen Substitutionen* $\Sigma \to 2^{\Sigma^*}$*, und* $\omega \in \Sigma^*$ *das Axiom ist.*

Wie im 0L Fall schreiben wir anstelle von $w \in \sigma_i(a)$ auch $a \to_{\sigma_i} w$, wobei—falls keine Mißverständnisse auftreten—σ_i auch weggelassen wird.

Definiton 3.4 *Es sei* $G = (\Sigma, H, \omega)$ *ein T0L-System,* $u \in \Sigma^*$ *und* $i \in \mathbb{N}_0$*. Wir definieren:*

$$L^i(G, u) = \begin{cases} \{u\} & i = 0 \\ \{v \in \Sigma^* \mid \exists x \in L^{i-1}(G, u) : \exists \sigma \in H : v \in \sigma(x)\} & \textit{sonst.} \end{cases}$$

Die von G *erzeugte Sprache wird mit* $L(G)$ *bezeichnet und ist definiert als*

$$L(G) = \bigcup_{i=0}^{\infty} L^i(G, \omega).$$

Nun führen wir noch ein paar Sprechweisen und Bezeichnung zum Thema Ableitungsrelation ein.

Definiton 3.5 *Es sei* G *ein [T]0L-System.*[1] *Wir definieren:*

1. *Wort* w_1 *leitet* w_2 *(in* G*)* direkt *ab (in Zeichen:* $w_1 \Rightarrow_G w_2$*), wenn gilt:* $w_2 \in L^1(G, w_1)$.
2. *Wort* w_1 *leitet* w_2 *(in* G*)* in n Schritten *ab (in Zeichen:* $w_1 \Rightarrow_G^n w_2$*), wenn gilt:* $w_2 \in L^n(G, w_1)$.

[1] Zur Vereinfachung führen wir eine Klammerschreibweise, z.B. [T]0L, bei L-Systemen ein, die andeuten soll, daß die entsprechende Beziehung sowohl beim Lesen des Klammerinhalts als auch beim Überlesen des Klammerinhalts gilt. Bei Gleichungen der Form X[Y][Z]0L = X'[Y][Z]0L wird sowohl auf der linken als auch auf der rechten Seite der jeweilige Klammerinhalt entweder gelesen oder überlesen. In dem gezeigten Fall ergeben sich aus der Gleichung X[Y][Z]0L = X'[Y][Z]0L die folgenden vier Beziehungen: X0L = X'0L, XY0L = X'Y0L, XZ0L = X'Z0L und XYZ0L = X'YZ0L.

3. *Wort w_1 leitet w_2 (in G)* echt *ab (in Zeichen: $w_1 \Rightarrow_G^+ w_2$), wenn gilt: $w_2 \in L^n(G, w_1)$ für ein $n \geq 1$.*

4. *Wort w_1 leitet w_2 (in G) ab (in Zeichen: $w_1 \Rightarrow_G^* w_2$), wenn gilt: $w_2 \in L^n(G, w_1)$ für ein $n \geq 0$.*

Falls keine Verwechslung möglich ist, wird auch $\Rightarrow$, $\Rightarrow^n$, $\Rightarrow^+$ und $\Rightarrow^*$ verwendet. Die von einem [T]0L-System G erzeugte Sprache kann jetzt durch

$$L(G) = \{ v \in \Delta^* \mid \omega \Rightarrow^* v \}$$

beschrieben werden.

Zwei wichtige Einschränkungen von [T]0L-Systemen sind deterministische und propagierende Systeme.

Definiton 3.6 *Es sei G ein [T]0L-System mit Alphabet Σ. Wir definieren:*

1. *G heißt* propagierend *oder P[T]0L-System, wenn für alle Substitutionen σ von G und für alle $a \in \Sigma$, $\lambda \notin \sigma(a)$ gilt.*

2. *G heißt* deterministisch *oder D[T]0L-System, wenn alle Substitutionen σ von G Homomorphismen sind.*

3. *Ist G sowohl propagierend als auch deterministisch, so ist es ein PD[T]0L-System.*

Die von G erzeugten Sprachen heißen entsprechend, d.h. P[T]0L-, D[T]0L- oder PD[T]0L-Sprache.

Das L-System aus Beispiel 1 ist propagierendes aber *nicht* deterministisches 0L-System, und das aus Beispiel 2 ist deterministisch aber *nicht* propagierend.

Beispiel 3 Das PD0L-System $G = (\{a, b\}, h, a)$ mit $h(a) = b$ und $h(b) = ab$ erzeugt die Sprache

$$L = \{a, b, ab, bab, abbab, bababbab, \ldots\}.$$

Bei genauerem hinsehen — Beweis per Induktion — stellt man fest, daß folgende Beziehung gilt:

$$a \Rightarrow^* u_1 \Rightarrow u_2 \Rightarrow u_1 u_2.$$

Betrachtet man nämlich die Längen der Wörter in L, dann sind diese gerade die Fibonacci-Zahlen.

Die Aussagen des folgenden Theorems sind offensichtlich erfüllt.

Theorem 3.1 1. *Es sei G ein 0L-System mit Alphabet Σ. Für $n \in \mathbb{N}_0$ und $u_1, u_2, v_1, v_2 \in \Sigma^*$ gelte $u_1 \Rightarrow^n v_1$ und $u_2 \Rightarrow^n v_2$. Dann folgt $u_1 u_2 \Rightarrow^n v_1 v_2$.*

2. *Es sei G ein [T]0L-System mit Alphabet Σ.*

(a) *Für* $n \in \mathbb{N}_0$ *und* u_1, u_2 *und* $w \in \Sigma^*$ *gelte* $u_1 u_2 \Rightarrow^n w$. *Dann existieren* $v_1, v_2 \in \Sigma^*$ *mit* $w = v_1 v_2$, $u_1 \Rightarrow^n v_1$ *und* $u_2 \Rightarrow^n v_2$.

(b) *Für* $m, n \in \mathbb{N}_0$ *und* $u, v, w \in \Sigma^*$ *gelte* $u \Rightarrow^m v$ *und* $v \Rightarrow^n w$. *Dann folgt* $u \Rightarrow^{m+n} w$.

Im Gegensatz zu L-Systemen gilt im Chomsky-Fall eine etwas verschärfte Variante der ersten Aussage des obigen Theorems: Es sei G eine Chomsky-Grammatik mit Nonterminalmenge N und Terminalmenge T. Für $u_1, u_2, v_1, v_2 \in \Sigma^*$ gelte $u_1 \Rightarrow^* v_1$ und $u_2 \Rightarrow^* v_2$. Dann folgt $u_1 u_2 \Rightarrow^* v_1 v_2$. An einem kleinen Gegenbeispiel kann man sich verdeutlichen, daß diese Aussage für den 0L-Fall nicht gilt.

Zum Ausklang dieses Abschnitts beschäftigen wir uns mit Abschlußeingenschaften von 0L-Sprachen. Als Vorbereitung zum nachfolgenden Theorem hier ein nützliches Lemma.

Lemma 3.1 *Die endliche Sprache* $\{a, a^3\}$ *ist keine 0L-Sprache.*

Beweis. Annahme es gäbe ein 0L-System $G = (\Sigma, \sigma, \omega)$, welches die Sprache $\{a, a^3\}$ erzeugt. Ohne Einschränkung der Allgemeinheit können wir $\Sigma = \{a\}$ annehmen. Nun gibt es zwei Fälle die zu unterscheiden sind, entweder gilt $\omega = a \Rightarrow a^3$ oder $\omega = a^3 \Rightarrow a$. Im ersten Fall muß $a^3 \in \sigma(a)$ gelten. Dies führt sofort zu einem Widerspruch, denn dann ist auch a^9 ein Element der Sprache, wegen $\omega = a \Rightarrow a^3 \Rightarrow a^9$. Auch der zweite Fall führt zu einem Widerspruch. Sei also $\omega = a^3$ und $a^3 \Rightarrow a$. Dann ist aber $\{\lambda, a\} \subseteq \sigma(a)$ und damit auch λ ein Element der Sprache, da $\omega = a^3 \Rightarrow a \Rightarrow \lambda$ gilt. □

Es folgt die etwas niederschmetternde Aussage, daß 0L-Sprachen *sehr* schlechte Abschlußeigenschaften haben.

Theorem 3.2 *Die Familie der 0L-Sprachen ist eine Anti-AFL.*

Beweis. Wir müssen zeigen, daß die Familie der 0L-Sprache unter keiner der AFL-Operationen (Vereinigung, λ-freier Iteration, λ-freien Homomorphismus, inversem Homomorphismus und Schnitt mit regulären Mengen) abgschlossen sind.

1. Wir zeigen, daß die Familie der 0L-Spracehn nicht unter Vereinigung abgeschlossen sind. Offenbar sind die endlichen Sprachen $\{a\}$ und $\{a^3\}$ auch 0L-Sprachen. Nach Lemma 3.1 ist aber $\{a\} \cup \{a^3\}$ keine 0L-Sprache.

2. Der Nichtabschluß von 0L unter λ-freier Iteration wird anhand der λ-freien Sprache
$$L = \{a^2\} \cup \{\, b^{2n} \mid n \geq 2 \,\}$$
gezeigt, die offenbar von dem D0L-System
$$G = (\{a, b\}, h, a^2) \quad \text{mit} \quad h(a) = h(b) = b^2$$
erzeugt wird. Wir nehmen an, daß L^+ eine 0L-Sprache ist. Dann existiert ein L^+ erzeugendes 0L-system $G' = (\{a, b\}, h, \omega)$. Da L^+ λ-frei ist,

muß $\lambda \notin h(a)$ und $\lambda \notin h(b)$ gelten. Folglich ist G' ein P0L-System. Die kürzesten Wörter von L^+ sind

$$a^2, a^4, b^4, a^6, a^2b^4 \quad \text{und} \quad b^4a^2.$$

Andere Wörter der Sprache sind länger. Da G' propagierend ist, muß das kürzeste Wort das Axiom sein, also $\omega = a^2$. Wir schließen $b \notin h(a)$, da anderenfalls die Ableitung $a^2 \Rightarrow_{G'} b^2 \notin L^+$ zum Widerspruch führt. Damit trotzdem Wörter erzeugt werden können, die nur aus Symbolen b bestehen, muß es ein $i > 1$ mit einer Produktion $b^i \in h(a)$ geben. Daraus folgt $a \notin h(a)$ und $a^3 \notin h(a)$, da anderenfalls die Ableitungen $a^2 \Rightarrow_{G'} ab^i \notin L^+$ bzw. $a^2 \Rightarrow_{G'} a^3b^i \notin L^+$ zum Widerspruch führen. Schließlich nehmen wir $a^2 \in h(a)$ an. Wegen $a^2 \Rightarrow^*_{G'} b^4 \in L^+$ existiert eine Produktion $b^i \in h(a)$ mit $i = 2$ oder $i = 3$. Dabei kommen $i = 0$ und $i = 1$ wegen der vorgergehenden Überlegung nicht in Frage, und $i = 4$ ist in dieser Ableitung wegen $\lambda \notin h(a)$ nicht möglich. Mit $a^2, b^i \in h(a)$ ergibt sich die Ableitung $a^2 \Rightarrow_{G'} a^2b^i \notin L^+$ und damit ein Widerspruch. Zusammenfassend stellen wir fest, daß $\lambda, a, a^2, a^3, b \notin h(a)$ und $\lambda, b \notin h(b)$ gelten.

Es wird nun gezeigt, daß das Wort $a^2b^4 \in L^+$ nicht gemäß G' erzeugbar ist. Hierzu betrachten wir alle Möglichkeiten, a^2b^4 direkt aus einem anderen Wort der Sprache zu erzeugen. Dafür kommen nur die (oben aufgeführten) Wörter der Länge kleiner gleich 6 in Frage.

(a) Aus $a^2 \Rightarrow_{G'} a^2b^4$ und dem oben bewiesenen Ausschluß von möglichen Produktionen folgt, daß eine Zahl $i \in \{1, 2\}$ existiert mit $a^2b^i \in h(a)$ und $b^{4-i} \in h(a)$. Dann erhalten wir jedoch auch die Ableitung $a^2 \Rightarrow_{G'} a^2b^ia^2b^i \notin L^+$.

(b) Die Ableitung $a^4 \Rightarrow_{G'} a^2b^4$ führt analog zum obigen Fall zum Widerspruch.

(c) Aus $b^4 \Rightarrow_{G'} a^2b^4$ folgt, daß das erste, zweite, dritte bzw. vierte Vorkommen von b in b^4 durch aw_1, w_2, w_3 bzw. w_4b durch geeignete Wörter w_1, w_2, w_3 bzw. w_4 mit $|w_2|, |w_3| \geq 1$ ersetzt werden muß. Die entsprechende Produktionen können auch in einer anderen Reihenfolge verwendet werden, man erhält so eine Ableitung $b^4 \Rightarrow_{G'} w_2w_3aw_1w_4b = v$ mit einem Wort $v \in L^+$ der Länge 6. Das einzige Wort aus L^+ mit diesen Eigenschaften ist jedoch $v = a^2b^4$. Dies steht im Widerspruch dazu, daß w_2w_3a mit $|w_2w_3| \geq 2$ ein Präfix von v ist.

(d) Die Ableitungen $a^6 \Rightarrow_{G'} a^2b^4$ und $b^4a^2 \Rightarrow_{G'} a^2b^4$ erfordern beide $b \in h(a)$, was jedoch bereits ausgeschlossen werden konnte.

Das Wort $a^2b^4 \in L^+$ ist also nicht gemäß G' erzeugbar.

3. Die Familie der 0L-Sprachen ist nicht unter Schnitt mit regulären Mengen abgeschlossen. Das 0L-System $G = (\{a\}, \{a \to a^2\}, a)$ erzeugt die Sprache $\{ a^{2^n} \mid n \geq 0 \}$. Schneidet man diese Sprache mit der regulären Menge $\{a, a^4\}$ erhält man die wieder die Sprache $\{a, a^4\}$, welche keine 0L-Sprache ist. Dies läßt sich analog zu Lemma 3.1 beweisen.

4. Der Nichtabschluß der Familie der 0L-Sprachen unter λ-freien Homomorphismus ergibt sich folgt: Sei $G = (\{a\}, \sigma, a^2)$ ein 0L-System mit $\sigma(a) = \{\lambda, a\}$. Offensichtlich erzeugt G die Sprache $\{\lambda, a, a^2\}$. Definiere nun einen λ-freien Homomorphismus $h(a) = a^5$. Dann gilt

$$h(\{\lambda, a, a^2\}) = \{\lambda, a^5, a^{10}\}.$$

Ähnlich wie in Lemma 3.1 kann man dann zeigen, daß $\{\lambda, a^5, a^{10}\}$ keine 0L-Sprache ist.

5. Der Nichabschluß der Familie der 0L-Sprachen unter inversem Homomorphismus kann z.B. anhand des Homomorphismus $h(a) = a^2$ und der 0L-Sprache $\{a\}$ gezeigt werden, da $h^{-1}(\{a\}) = \emptyset$ gilt, und die leere Menge keine 0L-Sprache ist. □

OK ←

3.2 L-Systeme mit Hilfssymbolen

Datum: 24. April
Datum: 6. Mai

Wie wir gesehen haben untercheidet man bei L-Systemen nicht zwischen Terminalen und Nichtterminalen wie bei Chomsky-Grammatiken. Daraus resultiert z.B., daß 0L-Sprachen *nicht* unter Vereinigung abschlossen sind. Man erweiterete daher L-Systeme zu sogenannten *extended* L-Systemen (im weiteren sprechen wir von L-Systemen mit Hilfssymbolen), die Terminale und Nichtterminale zu unterscheiden vermögen. Man beachte, daß bei L-Systemen auch Terminale Zeichen Worte ableiten können!

Definiton 3.7 *Ein* E[T]0L-System *ist ein 4-Tupel* $G = (\Sigma, H, \omega, \Delta)$*, falls* $\bar{G} = (\Sigma, H, \omega)$ *ein [T]0L-System ist und* $\Delta \subseteq \Sigma$ *gilt. Die Menge* Δ *wird dann als* Terminalalphabet *bezeichnet.*

Die von G *erzeugte Sprache wird mit* $L(G)$ *bezeichnet und ist definiert als*

$$L(G) = L(\bar{G}) \cap \Delta^*.$$

Die bei [T]0L-Systemen eingeführten Einschränkungen D und P übertragen sich in natürlicher Weise auch auf E[T]0L-Systeme in dem man diese Eigenschaft von dem zugrundeliegenden [T]0L-System fordert. Wir sprechen dann von ED[T]0L-, E[P][T]0L- und EPD[T]0L-Systemen und Sprachen.

Beispiel 4 Gegeben sei das EP0L-System

$$G = (\{X, Y, Z, A, B, C, a, b, c, F\}, \sigma, XYZ, \{a, b, c\})$$

mit der Produktionenmenge:

$$\begin{array}{lcl} X \to AX \mid a & \quad & A \to A \mid a \\ Y \to BY \mid b & & B \to B \mid b \\ Z \to CZ \mid c & & C \to C \mid c \end{array} \qquad \begin{array}{lcl} a \to F \\ b \to F \\ c \to F \\ F \to F. \end{array}$$

Wie man sich relativ schnell vergewissert, erzeugt G die *nicht* kontextfreie Sprache $\{ a^n b^n c^n \mid n \geq 1 \}$, denn für $i \geq 0$ findet man folgende Ableitung in G:

$$XYZ \Rightarrow^* A^i X b^i Y c^i Z \Rightarrow a^{i+1} b^{i+1} c^{i+1} \Rightarrow F^{3\cdot(i+1)}.$$

Man beachte, daß das gegebene System *nicht* deterministisch ist, da z.B. für X zwei Produktionen mit verschiedenen rechten Seiten existiert.

In der Literatur gibt es noch eine ganze Menge von Einschränkung und Erweiterung von L-Systemen. Bei der Benennung und Klassifizierung L-Systemen hat sich in der Literatur ein „orthogonales Baukastenprinzip'ëingebürgert, das Einschränkungen, Verallgemeinerungen und Alternativen enthält. Dabei unterscheidet man folgende Symbolarten:

A letztes Symbol: Ableitungsart

B vorletztes Symbol: kontextfreie versus kontextsensitive Ableitung

C links von **B**: Einschränkung oder Verallgemeinerung der Ableitungsfähigkeit.

D links von **C**: Art, wie aus der Satzformsprache die „endgültige“ Sprache gewonnen wird.

Anbei eine (nahezu komplette) Liste möglicher Symbole, die zur Klassifizierung dienen:

D	**C**	**B**	**A**	Beschreibung
A				**A**dult
			B	**B**harath (indisch für parallel)
C				**C**oding
	D			**D**eterministic—Homomorphismen anstatt Substitutionen
	DD			**D**oubly deterministic
	D			**D**ynamic deterministic
E				**E**xtension—Terminale und Nichtterminale
	F			**F**inite—mehrere, aber endlich viele Axiome
	G			**G**rowing—propagierend und keine Kettenproduktionen
H				**H**omomorphism
		I		Kontextsensitive Ableitungen
	J			**J**akut (finnisch für fragmentation)
	K			**K**akuizuteki (japanisch für stochastic)
			L	**L**indenmayer—parallele Ableitung
M				**M**ortal
			N	**N**ormal
		O		kontextfrei (eigentlich 0)
	P			**P**ropagating—keine λ-Produktionen
			R	**R**egular
	S			**S**ymmetric
			S	**S**equential—Ableitung wie im Chosmky Fall
	T			**T**able—mehrere Regelsätze
U				**U**nary—Alphabet ist einelementig

Mithilfe des Baukastenprinzips lassen sich dann Fragen wie APF0L $\stackrel{?}{=}$ KEFIR stellen (im übrigen sind die zwei Sprachklassen *nicht* äquivalent). OK ←

3.3 Abschlußeigenschaften von L-Systeme mit Hilfssymbolen

Datum: 15. Mai
Datum: 22. Mai

Bevor wir Abschlußeigenschaften von L-Systemen mit Hilfssymbolen untersuchen können, benötigen wir eine Normalform für E0L- und ET0L-Systeme. Ein E0L-System, das wie das System aus Beispiel 4 alle Symbole eines Terminalwortes auf einen Schlag erzeugt, wird synchronisiertes System genannt. Allgemein definieren wir:

Definiton 3.8 *Ein E[T]0L-System $G = (\Sigma, H, \omega, \Delta)$ heißt* synchronisiert*, wenn für alle $a \in \Delta$ und $w \in \Sigma^*$ mit $a \Rightarrow^+ w$ die Beziehung $w \notin \Delta^*$ folgt.*

Theorem 3.3 *Es existiert ein Algorithmus, der zu jedem beliebigen E[T]0L-System G ein äquvalentes synchronisiertes E[T]0L-System G' konstruiert. Dabei ist G' propagierend, wenn auch G propagierend ist.*

Beweis. Wir beweisen die Aussage nur für E0L-Systeme. Für ET0L-Systeme kann ein ähnlicher Beweis geführt werden.

Es sei $G = (\Sigma, \sigma, \omega, \Delta)$ ein E0L-System mit $\omega \notin \Delta^*$. Sollte dies nicht der Fall sein, so führen wir ein neues Zeichen $S \notin \Delta$ ein und setzen $\sigma(S) = \{\omega\}$. Wir definieren zuerst einen Homomorphismus g vermöge $g(a) = \bar{a}$ für $a \in \Sigma$ setzen $\bar{\Sigma} = \{\, g(a) \mid a \in \Sigma \,\}$. Dabei soll $\bar{\Sigma} \cap \Delta = \emptyset$ gelten. Nun konstruieren wir ein E0L-System $G' = (\bar{\Sigma} \cup \Delta \cup \{F\}, \sigma', g(\omega), \Delta)$, wobei F ein weiteres Symbol mit $F \notin \bar{\Sigma} \cup \Delta$ ist und die Produktionenmenge durch

$$\{\, g(a) \to g(w) \mid w \in \sigma(a) \,\} \cup \{\, g(a) \to a \mid a \in \Delta \,\} \cup \{\, a \to F \mid a \in \Delta \,\} \cup \{F \to F\}$$

bestimmt wird. Offenbar ist G synchronisiert. Die Ableitungen gemäß G_1 werden in G in gequerter Form simuliert, wobei im letzten Schritt die Terminalsymbole eingeführt werden. Es gilt $L(G) = L(G_1) = L(G')$. Wir erkennen sofort, daß G propagierend ist, wenn es auch G' ist. □

Nun können wir uns den Abschlußeigenschaften von L-Systemen mit Hilfssymbolen zuwenden. Unglücklicher Weise müßen wir beim folgenden Theorem auf Ergebnisse verweisen die erst in den nächsten Abschnitten gezeigt werden und zusätzlich brauchen wir folgende Normalform für E[P]T0L-Systeme.

Lemma 3.2 *Zu jedem E[P]T0L-System mit k Tafeln kann man ein äquivalentes E[P]T0L-System mit* zwei *Tafeln konstruieren.*

Beweis. Sei $G = (\Sigma, \{\sigma_1, \ldots, \sigma_k\}, S, \Delta)$ ein E[P]T0L-System. Wir konstruieren ein äquivalentes E[P]T0L-System $G' = (\Sigma', \{\sigma'_1, \sigma'_2\}, [S, 0], \Delta)$ wie folgt. Zuerst definieren wir das neue Alphabet Σ' wie folgt:

$$\Sigma' = \Sigma \cup \{\, [a, i] \mid a \in \Sigma \text{ und } 0 \le i \le k \,\} \cup \{F\},$$

wobei F ein neues Symbol mit $F \notin \Sigma' \setminus \{F\}$ ist. In der ersten Tafel von G' wird zwischen den einzelnen $[a,i]$ „hin und her“ gewechselt. Um etwas genauer zu sein:

$$\sigma_1'([a,i]) = \begin{cases} [a,i+1] & \text{falls } 0 \leq i < k \\ [a,0] & \text{sonst.} \end{cases}$$

In der zweiten Tafel wird die eigentliche Regelanwendung simuliert und evetuell terminiert, d.h. wir nehmen die folgenden Regeln auf:

$$\sigma_2'([a,i]) = \begin{cases} \bigcup_{w \in \sigma_i(a)} h_i(w) & \text{falls } 1 \leq i \leq k \\ a & \text{sonst,} \end{cases}$$

wobei für jedes $0 \leq i \leq k$, $h_i : \Sigma \to \Sigma'$ ein Homomorphismus mit $h_i(a) = [a,i]$ ist.

Um die Konstruktion zu vervollständigen nehmen wir in σ_1' und σ_2' noch die Produktionen $a \to F$ und $F \to F$ auf.

Eine Ableitung in G' startet mit dem Wort $[S,0]$. Um die Regeln aus der ursprünglichen Tafel σ_i anzuwenden muß man zuerst σ_1' wiederholt anwenden, um auf $[S,i]$ zu kommen. Mit σ_2' kann man dann die ursprüngliche Tafel simulieren und erhält so ein Wort aus der Menge $h_i(\sigma_i(S))$. Um nun erneut eine Tafelanwendung aus G zu simulieren, muß mit σ_1' wieder das entsprechend „gefärbte Wort“ erzeugt werden. Will man terminieren, so kann man das nur durch Anwendung von σ_2', wenn die Satzform nur aus Elementen der Form $[a,0]$, für $a \in \Sigma$, besteht.

Mit dieser Vorrede kann man sich nun leicht per Induktion überzeugen, daß das so konstruierte System G' äquivalent zum umsprünglichen System G ist. □

Theorem 3.4 *Die Familie ET0L-Sprachen ist eine volle AFL.*

Beweis. Es seien L_1 und L_2 ET0L-Sprachen. Dann existieren ET0L-Systeme G_1' und G_2' mit $L(G_1') = L_1$ und $L(G_2') = L_2$. Nach Theorem 3.7 gibt es EPT0L-Systeme $G_1 = (\Sigma_1, H_1, S_1, \Delta_1)$ und $G_2 = (\Sigma_2, H_2, S_2, \Delta_2)$ mit jeweils zwei Tafeln und mit $L(G_1) = L_1 \setminus \{\lambda\}$ und $L(G_2) \setminus \{\lambda\}$. Insbesondere gilt also $S_1 \in \Sigma_1 - \Delta_1$ und $S_2 \in \Sigma_2 - \Delta_2$. Nach Theorem 3.10 ist entscheidbar, ob $\lambda \in L(G_i)$, für $i = 1,2$, erfüllt ist. Für $\lambda \in L_i$ wird zusätzlich die Produktion $S_i \to \lambda$ in jeder Tafel von H_i aufgenommen. Ohne Beschränkung der Allgemeinheit gelte

$$(\Sigma_1 \setminus \Delta_1) \cap \Sigma_2 = \emptyset \quad \text{und} \quad (\Sigma_2 \setminus \Delta_2) \cap \Sigma_1 = \emptyset$$

was zu $\Sigma_1 \cap \Sigma_2 = \Delta_1 \cap \Delta_2$ äquivalent ist. Es sei S ein neues Symbol.

1. Die Familie der ET0L-Sprachen ist abgeschlossen unter Vereinigung. Dies zeigen wir, indem wir ein ET0L-System

   $$G = (\Sigma, H, S, \Delta_1 \cup \Delta_2)$$

   mit $L(G) = L_1 \cup L_2$ konstruieren. Wir setzen

   $$\Sigma = \Sigma_1 \cup \Sigma_2 \cup \{S\} \quad \text{und} \quad H = \{\sigma_0\} \cup \{\, \hat{\sigma} \mid \sigma \in H_1 \,\} \cup \{\, \tilde{\sigma} \mid \sigma \in H_2 \,\},$$

wobei die Tafeln σ_0, $\hat{\sigma}$ für $\sigma \in H_1$ und $\tilde{\sigma}$ für $\sigma \in H_2$ durch

$$\sigma_0(a) = \begin{cases} \{a\} & \text{falls } a \in \Sigma_1 \cup \Sigma_2 \\ \{S_1, S_2\} & \text{sonst,} \end{cases} \qquad \hat{\sigma}(a) = \begin{cases} \sigma(a) & \text{falls } a \in \Sigma_1 \\ \{a\} & \text{sonst} \end{cases}$$

und

$$\tilde{\sigma}(a) = \begin{cases} \sigma(a) & \text{falls } a \in \Sigma_2 \\ \{a\} & \text{sonst} \end{cases}$$

definiert werden. Da durch die Produktionen von G_1 und G_2 Terminalzeichen nicht verändert werden und die übrigen Zeichen der beiden Systeme disjunkt sind, kann nach Einführung des Zeichens $S_i, i = 1, 2$ nur eine Ableitung gemäß G_i simuliert werden. Ein Übergang von z.B. G_1 nach G_2 ist nicht möglich. Damit erhalten wir $L(G) = L_1 \cup L_2$.

2. Der Abschluß unter λ-freier Iteration wird mit Hilfe des ET0L-Systems $G = (\Sigma, H, S, \Delta_1)$ mit $\Sigma = \Sigma_1 \cup \{S\}$ und $H = \{\sigma_0\} \cup \{\, \hat{\sigma} \mid \sigma \in H_1 \,\}$ bewiesen, wobei die Tafeln σ_0 und $\hat{\sigma}$ für $\sigma \in H_1$ durch

$$\sigma_0(a) = \begin{cases} \{a\} & \text{für } a \in \Sigma_1 \\ \{S, S^2, S_1\} & \text{sonst} \end{cases}$$

und

$$\hat{\sigma}(a) = \begin{cases} \sigma(a) & \text{für } a \in \Sigma_1 \\ \{S\} & \text{sonst} \end{cases}$$

definiert sind.

3. Für den Nachweis des Abschlusses unter Durchschnitt mit regulären Sprachen wird ein deterministischer endlicher Automat $M = (\Sigma, Q, \delta, q_0, F)$ betrachtet. Mit Hilfe der Menge

$$\Theta = \{\, [p, a, q] \mid p, q \in Q \text{ und } a \in \Sigma_1 \,\}$$

und der neuen Symbole F und S wird zunächst das Alphabet $\Theta \cup \Delta_1 \cup \{F, S\}$ definiert. Für jede Tafel $\sigma \in H_1$ wird eine Tafel $\hat{\sigma}$ durch

$$\begin{aligned} \hat{\sigma}([p,a,q]) &= \{\, [p, a_1, q_{i_1}][q_{i_1}, a_2, q_{i_2}] \dots [q_{i_n - 1}, a_n, q] \mid a_1 \dots a_n \in \sigma(a) \text{ und} \\ &\qquad q_{i_1}, \dots, q_{i_n - 1} \in Q \,\} \quad \text{für } [p, a, q] \in \Theta, \\ \hat{\sigma}(S) &= \{\, [q_0, S_1, q_f] \mid q_f \in F \,\} \cup \{\, \lambda \mid \lambda \in L(G_1) \text{ und } q_0 \in F \,\} \quad \text{und} \\ \hat{\sigma}(a) &= \{a\} \quad \text{für } a \in \Delta_1 \cup \{F\} \end{aligned}$$

bestimmt. Man beachte dabei, daß eine eventuell vorhandene Produktion $S \to \lambda$ in $\hat{\sigma}([p, a, q])$ nicht berücksichtigt wird, die zugehörige Produktion jedoch in $\hat{\sigma}(S)$ aufgenommen wird. Außerdem wird eine finale Tafel σ_{fin} durch

$$\sigma_{fin}([p, a, q]) = \begin{cases} \{a\} & \text{falls } a \in \Sigma \cap \Delta_1 \text{ und } \delta(p, a) = q \\ \{F\} & \text{sonst} \end{cases}$$

für $[p, a, q] \in \Theta$ und $\sigma_{fin}(a) = \{F\}$ für alle anderen Symbole im Alphabet. Mit

$$H = \{\sigma_{fin}\} \cup \{\, \hat{\sigma} \mid \sigma \in H_1 \,\}$$

erhalten wir ein ET0L-System $G = (\Theta \cup \Delta_1 \cup \{F, S\}, H, S, \Delta_1 \cap \Sigma)$. Die Tafel σ_{fin} sorgt am Abschluß einer Ableitung dafür, daß genau die Wörter aus $L(G_1) \cap L(M)$ durch G erzeugt werden. Die verwendete Konstruktion wird in der Literatur auch als „Tripelkonstruktion“ bezeichnet.

4. Wir betrachten den Abschluß unter Homomorphismus. Es sei $\varphi : \Delta_1^* \to \Theta^*$ ein Homomorphismus. Ohne Einschränkung können wir annehmen, daß $\Sigma_1 \cap \Theta = \emptyset$ gilt. Sei $F \notin \Sigma \cup \Theta$ ein neues Symbol. Definiere für jede Tafel σ aus G_1 eine neue Tafel $\hat{\sigma}$ wie folgt:

$$\hat{\sigma}(a) = \begin{cases} \sigma(a) & \text{für } a \in \Sigma_1 \\ \{F\} & \text{für } a \in \Theta \cup \{F\} \end{cases}$$

Darüberhinaus definieren wir noch eine Tafel σ_T mit

$$\sigma_T(a) = \begin{cases} \{\varphi(a)\} & \text{für } a \in \Delta_1 \\ \{F\} & \text{für } a \in (\Sigma_1 \setminus \Delta_1) \cup \Theta \cup \{F\}. \end{cases}$$

Setzen wir $G = (\Sigma \cup \Theta \cup \{F\}, \hat{H}, S_1, \Theta)$ mit $\hat{H} = \{\, \hat{\sigma} \mid \sigma \in H_1 \,\} \cup \{\sigma_T\}$ so folgt $L(G) = \varphi(L_1)$. Wir bemerken, daß im Falle einer vorhandenen Produktion $S_1 \to \lambda$ das Wort $\lambda \in \varphi(L(G_1))$ durch eine Tafel $\hat{\sigma}$ mit $\sigma \neq \sigma_T$ erzeugt wird.

5. Für den Nachweis des Abschlusses unter inversem Homomorphismus wird ein Homomorphismus $\varphi : \Delta^* \to \Delta_1^*$ betrachtet. Dabei können wir ohne Beschränkung der Allgemeinheit annehmen, daß $\Delta \cap \Sigma_1 = \emptyset$ gilt. Wir definieren eine Menge $K \subseteq (\Delta \cup \Delta_1)^*$ durch

$$K = \{\, y = z_0 x_1 z_1 x_2 z_2 \dots x_k z_k \in (\Delta \cup \Delta_1)^* \mid x_1 \dots x_k \in L(G_1), \\ x_1, \dots, x_k \in \Delta_1,\ z_0, \dots, z_k \in \Delta^* \text{ und } k \in \mathbb{N}_0 \,\}.$$

Dann konstruieren wir ein ET0L-System $G = (\Sigma, \hat{H}, S_1, \Delta_1 \cup \Delta)$ mit $L(G) = K$ wie folgt. Es wird

$$\Sigma = \Sigma_1 \cup \Delta \quad \text{und} \quad \hat{H} = \{\sigma_0\} \cup \{\, \hat{\sigma} \mid \sigma \in H_1 \,\}$$

gesetzt, wobei

$$\begin{aligned} \sigma_0(a) &= \{\, xay \mid x, y \in \Delta \cup \{\lambda\}\} \quad \text{für } a \in \Delta_1, \\ \sigma_0(a) &= \{a\} \quad \text{für } a \in (\Sigma_1 \setminus \Delta_1) \cup \Delta \text{ und } a \neq S_1, \\ \sigma_0(S_1) &= \{\, xS_1y \mid x, y \in \Delta \cup \{\lambda\} \,\} \end{aligned}$$

und $\hat{\sigma}(a) = \sigma(a)$ für $a \in \Sigma_1$ sowie $\hat{\sigma}(a) = \{a\}$ für $a \in \Delta$ für alle $\sigma \in H_1$ gilt. Mit Hilfe der Tafel $\hat{\sigma}$ erzeugt G zunächst $L_1 = L(G_1)$, worauf durch wiederholte Anwendung von σ_0 die z_i „eingestreut“ werden. Aufgrund der Eigenschaften von G_1 hat eine spätere Anwendung von $\hat{\sigma}$ keine Wirkung. $\sigma_0(S_1)$ wird benötigt, um auch im Fall $\lambda \in L_1$ ein $z_0 \in K$ zu erhalten.

Als nächstes wird die Menge

$$M = \{\, \varphi(y_1)y_1\varphi(y_2)y_2 \dots \varphi(y_n)y_n \mid n \geq 0 \text{ und } y_i \in \Delta \,\}$$

betrachtet. Sie ist regulär, da sie von einer Grammatik mit den Produktionen $S \to S\varphi(y)y$ für alle $y \in \Delta$ sowie $S \to \lambda$ erzeugt wird. Offenbar gilt:

$$K \cap M = \{\, \varphi(y_1)y_1\varphi(y_2)y_2 \dots \varphi(y_n)y_n \mid y_i \in \Delta, \text{für } 1 \leq i \leq n, \text{ und} \\ \varphi(y_1) \dots \varphi(y_n) \in L(G_1) \,\}.$$

Das bedeutet, daß für $y = y_1 \dots y_n$ mit $y_1, \dots, y_n \in \Delta$ die Äquivalenz $\varphi(y) \in L(G_1)$ genau dann, wenn $\varphi(y_1)y_1 \dots \varphi(y_n)y_n \in K \cap M$ erfüllt ist. Wird jetzt ein Homomorphismus $\psi : (\Delta \cap \Delta_1)^* \to \Delta$ durch

$$\psi(a) = \begin{cases} a & \text{für } a \in \Delta \\ \lambda & \text{für } a \in \Delta_1 \end{cases}$$

definiert, so erhalten wir $\psi(K \cap M) = \varphi^{-1}(L(G_1))$. Da $K \in$ ET0L und M regulär ist, folgt $K \cap M \in$ ET0L, woraus sich die Beziehung $\psi(K \cap M) = \varphi^{-1}(L(G_1)) \in$ ET0L ergibt. □

Mit ähnlichen Techniken kann man zeigen, daß E0L- und EDT0L-Sprachen auch relativ gute Abschlußeigenschaften haben.

Theorem 3.5 *Die Familie der E0L- und EDT0L-Sprachen sind unter den AFL-Operation, mit Ausnahme von inversen Homomorphismen, abgeschlossen.* □

Der interessierte Leser wir sicher schnell verifizieren, daß z.B. ED0L-Sprachen *nicht* unter Vereinigung und Konkatenation abschlossen sind. OK ←

3.4 Zur Rolle von λ-Produktionen in L-Systemen mit Hilfssymbolen

Datum: 27. Mai

Bei regulären und kontextfreien Chomsky-Grammatiken erhöhen λ-Produktionen die generative Mächtigkeit (modulo dem leeren Wort) nicht. Genauer gesagt, kann man für jede reguläre (kontextfreie) Chomsky-Grammatik G eine λ-freie Grammatik G' vom selbem Typ konstruieren so, daß $L(G') = L(G) \setminus \{\lambda\}$ gilt. Wir wollen nun die Rolle von λ-Produktionen bei L-Systemen genauer betrachten.

Für den einfachsten Fall bei L-Systemen mit Hilfssymbolen, nämlich ED0L-Sprachen, erhöhen λ-Produktionen die generative Mächtigkeit.

Theorem 3.6 *EPD0L $\subset$ ED0L.*

Beweis. Die endliche Sprache $L = \{a^2, a^2b^2\}$ wird von dem D0L-System $G = (\{a, b\}, h, a^2b^2)$ mit $h(a) = a$ und $h(b) = \lambda$ erzeugt. Wir zeigen, daß es kein EPD0L-System G' gibt mit $L = L(G')$. Annahme, es gäbe ein solches System, dann ergibt sich aufgrund der P-Eigenschaft, daß

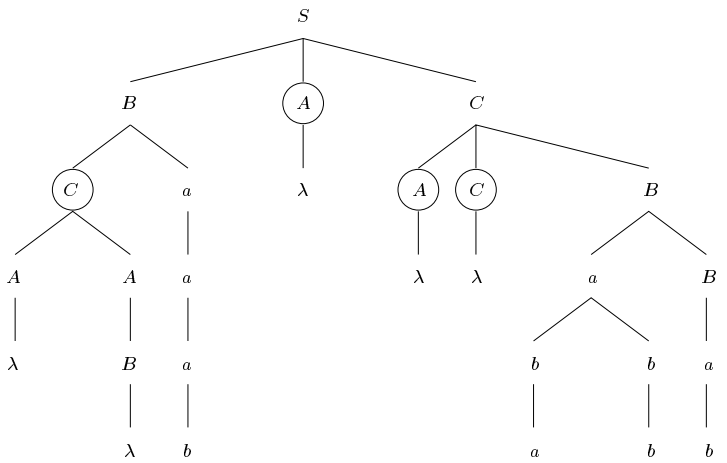
$$a^2 \Rightarrow^* a^2b^2$$

gelten muß. Da G' zusätzlich ein deterministisches System ist gilt: Wenn $a \Rightarrow^n u_1$ und $a \Rightarrow^n u_2$, dann ist $u_1 = u_2$. Also kann man aus a^2 in G' nur Worte der Form uu ableiten. Da a^2b^2 nicht von dieser Form ist, kann es kein System G' mit $L = L(G')$ geben. □

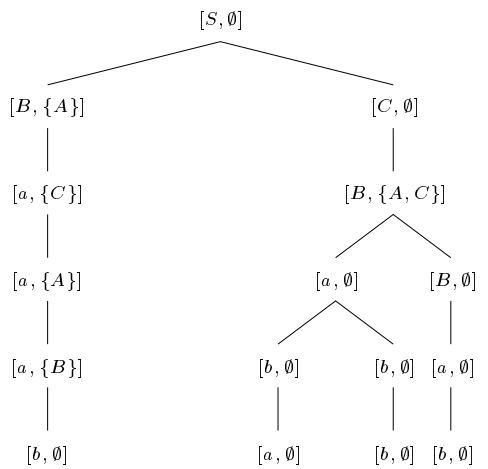
Demgegenüber findet man bei E0L-, EDT0L- und ET0L-Sprachen das gleiche Szenario wie bei regulären und kontextfreien Chomsky-Sprachen.

Theorem 3.7 *EP0L = E0L und EP[D]T0L = E[D]T0L.*

Beweis. Es sei $H = (\Sigma, h, \omega, \Delta)$ ein E0L-System. Falls $L(H)$ endlich ist, ist die Aussage offensichtlich erfüllt, da jede endliche Menge eine E0L-Sprache ist. Im folgenden nehmen wir daher an, daß $L(H)$ unendlich ist, wobei wir $\omega = S \in \Sigma \setminus \Delta$ voraussetzen können. Zunächst wollen wir die intuitive Beweisidee verdeutlichen. Wir gehen von einem Ableitungsbaum in H für das Wort bab^2 aus.



Dieser Ableitungsbaum soll in G derart simuliert werden, daß die von den eingekreisten Knoten ausgehenden Teilbäume nicht vorkommen, da diese Knoten unproduktive Vorkommen repräsentieren. Die wegzulassenden Teilbäume werden sich zunächst dadurch gemerkt, daß die zugehörigen unproduktiven Vorkommen in den jeweiligen Schritten als zweite Komponente der jeweiligen Marken der produktiven Vorkommen notiert werden. Man merkt sich also die Symbole, die man in das leere Wort ableiten möchte.



Erst in der letzten Stufe sind keine unproduktiven Vorkommen mehr vorhanden. Dann kann bab^2 durch einen weiter unten beschriebenen Synchronisationsmechanismus erzeugt werden. Da vorab nicht bekannt ist, ob ein Vorkommen unproduktiv ist, müssen alle Möglichkeiten solcher Bäume berücksichtigt werden.

Für ein Wort w über einem beliebigen Alphabet Σ bezeichne $alph(w)$ die Menge aller in w enhaltenen Symbole aus Σ.

Formal wird zu H das EP0L-System $G = (V, g, [S, \emptyset], \Delta)$ konstruiert. Wir setzen $V = V_1 \cup \{F\} \cup \Delta$ mit $V_1 = \{[a, Z] \mid a \in \Sigma, Z \subseteq \Sigma\}$. Die Substitution g wird wie folgt definiert:

1. Für alle $a \in \Sigma$ mit $b_1 \dots b_k \in h(a)$, $b_1, \dots, b_k \in \Sigma$, $k \geq 2$, und alle $Z \subseteq \Sigma$ erhalten wir
$$[b_{i_1}, Z_{i_1}][b_{i_2}, Z_{i_2}] \dots [b_{i_p}, Z_{i_p}] \in g([a, Z]),$$
falls die folgenden Eigenschaften erfüllt sind: Es gilt $1 \leq i_1 < i_2 < \dots < i_p \leq k$ und
$$\begin{aligned} Z_{i_1} &= alph(b_1 b_2 \dots b_{i_1-1}) \cup alph(b_{i_1+1} b_{i_1+2} \dots b_{i_2-1}), \\ Z_{i_j} &= alph(b_{i_j+1} b_{i_j+2} \dots b_{i_{j+1}-1}) \quad \text{für } 2 \leq j \leq p-1 \text{ sowie} \\ Z_{i_p} &= alph(b_{i_p+1} b_{i_p+2} \dots b_k) \cup Z' \end{aligned}$$
für p ≥ 2 und
$$Z_{i_1} = alph(b_1 b_2 \dots b_{i_1-1}) \cup alph(b_{i_1+1} b_{i_1+2} \dots b_{i_k}) \cup Z'$$
für $p = 1$ mit jeweils
$$Z' \in succ_H(Z) = \{\, U \subseteq \Sigma \mid \bigvee_{w_1, w_2 \in \Sigma^*} alph(w_1) = Z, \; alph(w_2) = U \text{ und } w_1 \Longrightarrow_H \omega_2 \,\}.$$

2. Für alle $a \in \Sigma$ mit $b \in h(a)$ und alle $Z \subseteq \Sigma$ gilt $[b, Z'] \in g([a, Z])$, falls $Z' \in succ_H(Z)$ gilt.
3. Für alle $a \in \Sigma$ mit $\lambda \in h(a)$ und alle $Z \subseteq \Sigma$ gilt $F \in g([a, Z])$.
4. Für alle $a \in \Delta$ gilt $a \in g([a, \emptyset])$.
5. Es gilt $g(F) = \{F\}$ und $g(a) = \{F\}$ für alle $a \in \Delta$.

Durch (4) und (5) wird die Synchronisation erreicht.

Bis auf das in den Sprachen E0L eventuell vorhandene leere Wort λ stimmen E0L und EP0L überein. Offensichtlich ist die Konstruktion des Beweises auch für ein gegebenes ET0L-System H richtig, wobei jeder Tafel von H genau eine Tafel von G zugeordnet wird. Somit ist das Theorem auch für ET0L-Systeme erfüllt. □

Abschließend sei noch erwähnt, daß man bei L-Systemen *ohne* Hilfssymbole in allen vier Fällen (D0L, 0L, DT0L und T0L) die entsprechenden propagierenden Varianten *echt* separieren kann. OK ←

3.5 L-Systeme mit Hilfssymbolen und die Chomsky Hierarchie

Datum: 5. Juni

In diesem Abschnitt soll die Beziehung von L-Systeme mit Hilfssymbolen und den Klassen der Chomsky Hierarchie geklärt werden. Zuerst ein offensichtliches Lemma, mit dem man L-Systeme mit Hilfssymbolen von L-Systeme ohne Hilfssymbole trennen kann.

Lemma 3.3 *Die leere Sprache $\emptyset$ ist in E0L und in ET0L enthalten, aber weder in 0L noch in T0L.*

Korollar 3.1 *$0L \subset E0L$ und $T0L \subset ET0L$.*

Nun zur Einordnung von L-Systeme mit Hilfssymbolen in die Chomsky Hierarchie und zu deren Erzeugungsmächtigkeit untereinander.

Theorem 3.8 *$CFL \subset E0L$.*

Beweis. Sei $G = (N, T, P, S)$ eine kontextfreie Grammatik. Wir konsturieren ein E0L-System $H = (N \cup T, \sigma, S, T)$ mit $\sigma(a) = \{a\}$ für alle $a \in T$ und $\sigma(A) = \{ u \mid A \to u \text{ ist in } P \}$ für alle $A \in N$.

Um die Inklusion $L(G) \subseteq L(H)$ zu zeigen, beweist man per Induktion $S \Rightarrow_G^* u \Rightarrow_G v$ impliziert $S \Rightarrow_H^* u \Rightarrow_H v$, und für den Fall $L(H) \subseteq L(G)$ zeigt man $S \Rightarrow_H^* u \Rightarrow_H v$ impliziert $S \Rightarrow_G^* u \Rightarrow_G^* v$.

Die Echtheit der Inklusion folgt z.B. aus Beispiel 4, da die Sprache $\{ a^n b^n c^n \mid n \geq 1 \}$ *nicht* kontextfrei ist. □

Für das nächste Theorem E0L $\subset$ ET0L benötigen wir noch ein Hilfslemma.

Definiton 3.9 *Sei $L \in \Sigma^*$ eine Sprache, $\Theta \subset \Sigma, \Theta \neq \emptyset$ und $w \in \Sigma^*$. Mit $pres_\Theta(w)$ bezeichnen wir das Wort, das sich durch entfernen aller Symbole ergibt, die nicht zu Θ gehören.*

Die Sprache L heißt Θ-determiniert, *wenn für alle $k \in \mathbb{N}$ eine Zahl $n_k \in \mathbb{N}$ existiert, so daß für alle $v, w \in L$ mit $|v|, |w| > n_k$ und $v = v_1 v' v_2, w = v_1 w' v_2$ mit $|v'|, |w'| < k$ die Beziehung $pres_\Theta(v) = pres_\Theta(w)$ gilt.*

Nun zum Hilfslemma, das wir ohne Beweis angeben, da dieser von sehr technischer Natur ist. Ein Beweis findet sich z.B. im Buch von Rozenberg und Salomaa [9] auf Seite 88.

Lemma 3.4 *Es sei $L \in E0L$ eine Θ-determinierte Sprache. Dann existieren $c, d \in \mathbb{N}$, so daß für jedes Wort $w \in L$ mit $|w|_\Theta > c$ die Beziehung $|w| < d^{|w|_\Theta}$ gilt. Hierbei sei $|w|_\Theta$ definiert als $|pres_\Theta(w)|$.* □

Theorem 3.9 *$E0L \subset ET0L$.*

Beweis. Um zu zeigen, daß die Inklusion echt ist, verwenden wir die Sprache $L = \{\, \$w\$w\$w \mid w \in \{a,b\}^* \,\}$, die durch das folgende ET0L-System $G = (\Sigma, H, \omega, \Delta)$ mit $\Sigma = \{S, A, B, \$, a, b\}$, $\Delta = \{\$, a, b\}$, $\omega = SSS$ und $H = \{h_1, h_2\}$, wobei die Tafel h_1 definiert ist durch

$$S \to A, A \to Sa, B \to \$ \quad \text{und } x \to x \text{ für } x \in \{\$, a, b\},$$

und h_2 durch

$$S \to B, A \to \$, B \to Sb \quad \text{und } x \to x \text{ für } x \in \{\$, a, b\},$$

erzeugt wird.

Mit der ersten Tafel wird also im Laufe der Ableitung an allen Teilen des Wortes ein a angehängt, mit der zweiten entsprechend ein b.

Die Ableitungen in G beginnen demnach wie folgt:

$$SSS \begin{cases} \overset{h_1}{\Longrightarrow} AAA & \begin{cases} \overset{h_1}{\Longrightarrow} SaSaSa \\ \overset{h_2}{\Longrightarrow} \$\$\$ \end{cases} \\ \overset{h_2}{\Longrightarrow} BBB & \begin{cases} \overset{h_1}{\Longrightarrow} \$\$\$ \\ \overset{h_2}{\Longrightarrow} SbSbSb \end{cases} \end{cases}$$

Um zu zeigen, daß L keine E0L-Sprache ist, verwenden wir Lemma 3.4. Zuerst zeigen wir, daß L $\{a\}$-determiniert ist.

Zu $k \in \mathbb{N}$ setzen wir $n_k = 3k + 3$. Betrachte nun $v = v_1 u v_2, w = v_1 u' v_2 \in L$ mit $|v|, |w| > 3k + 3$. Die Wörter v und w haben die Gestalt $v = \$w'\$w'\$w'$ bzw. $w = \$w''\$w''\$w''$. Aufgrund der Länge der Wörter muß gelten $|w'|, |w''| > k$. Da entweder v_1 oder v_2 ein komplettes Teilwort, das druch \$ begrenzt wird, enthalten muß, folgt $w' = w''$. Damit also $v = w$ und $\text{pres}_{\{a\}}(v) = \text{pres}_{\{a\}}(w)$.

Wäre L eine E0L-Sprache, dann existierten $c, d \in \mathbb{N}$, die Lemma 3.4 erfüllen. Wir wählen $n, m \in \mathbb{N}$ so, daß $n > \frac{c}{3}$ und $m = d^{3n}$. Betrachte das Wort $w = \$a^n b^m \$ a^n b^m \$ a^n b^m \in L$. Es gilt $|w|_a = 3n > c$. Andererseits ist aber $|w| = 3 + 3n + 3m > d^{3n} = d^{|w|_a}$, was im Widerspruch zu $|w| < d^{|w|_a}$ steht. □

Nachfolgend zeigen wir, daß die Familie der ET0L-Sprachen kontextsensitiv ist.

Theorem 3.10 *ET0L ⊂ CS.*

Beweis. Sei G ein ET0L-System mit Terminalalphabet Δ. Da die Konstruktion einer kontextsensitiven Grammatik für $L(G)$ selbst etwas mühsam ist, gehen wir folgenden Umweg. Offensichtlich gilt

$$L(G) = \bigcup_{a,b \in \Delta} a \cdot \delta(a, L(G), b) \cdot b \cup F,$$

wobei $\delta(a, L, b) = \{\, u \mid aub \in L \}$ und F eine endliche Menge ist. Da die Familie der ET0L-Sprachen eine AFL ist, ist sowohl jedes $\delta(a, L(G), b)$ für jedes $a, b \in \Delta$

als auch die endliche Menge F eine ET0L-Sprache. Es genügt nun zu zeigen, daß $a \cdot \delta(a, L, b) \cdot b$ kontextsensitiv ist.

Sei also $G_{ab} = (\Sigma, H, \omega, \Delta)$ ein ET0L-System mit $H = \{\sigma_1, \ldots, \sigma_n\}$. Ohne Einschränkung der Allgemeinheit können wir annehmen, daß G_{ab} ein propagierendes System ist.

Wir konstruieren eine kontextsenstive Grammatik $G'_{ab} = (N, \Delta, P, S)$ mit Nonterminalmenge

$$N = \{S, A, B\} \cup \{\, a' \mid a \in \Sigma \,\} \cup \{\, [a', T_i], [a', \sigma_i] \mid 1 \leq i \leq n \,\},$$

wobei alle Vereinigungen disjunkt sein sollen, für die Sprache $a \cdot L(G_{ab}) \cdot b$. Darüber hinaus definiere einen Homomorphismus $g : \Sigma^* \to N^*$ vermöge $g(a) = a'$ für alle $a \in \Sigma$.

Die Produktionenmenge P besteht aus vier Mengen $P_1, \ldots, P_4$. Die Simulation startet mit einer Regel aus P_1, welche wie folgt aussieht:

$$P_1 = \{S \to Ag(\omega)B\} \cup \{\, A \to a, B \to b\}.$$

Die $[a', T_i]$'s dienen dann als Symbole, die sich von links nach rechts fortpflanzen und Symbole der Form $[a', \sigma_i]$ zurücklassen, die die Anwendung der Tafel σ_i simulieren. Zuerst zu den Produktionen die $[a', T_i]$'s verwenden. Um die $[a', T_i]$'s zu erzeugen verwendet man die Regel aus

$$P_2 = \{\, Aa' \to A[a', T_i] \mid 1 \leq i \leq n \,\}.$$

Das Wandern von links nach rechts geschieht wie folgt

$$\begin{aligned} P_3 = \{\, [a', T_i]b' \to [a', \sigma_i][b', T_i] \mid 1 \leq i \leq n \,\} \\ \cup \{\, [a', T_i]B \to [a', \sigma_i]B \mid 1 \leq i \leq n \,\}. \end{aligned}$$

Die Terminierung wird mit den Produktionen aus P_4 realisiert. Wir definieren

$$\begin{aligned} P_4 = \{\, [a', \sigma_i] \to g(u) \mid 1 \leq i \leq n \text{ und } u \in \sigma_i(a) \,\} \\ \cup \{\, [a', \sigma_i] \to u \mid 1 \leq i \leq n,\, u \in \sigma_i(a) \text{ und } u \in \Delta^* \,\}. \end{aligned}$$

Dann zeigt man per Induktion, daß $\omega \Rightarrow^* u \Rightarrow_{\sigma_i} v$ genau dann, wenn $S \Rightarrow^* Ag(u)B \Rightarrow^* Ag(v)B$ oder $S \Rightarrow^* Ag(u)B \Rightarrow^* avb$ falls $v \in \Delta^*$. Damit wäre gezeigt, daß $L(G'_{ab}) = a \cdot L(G_{ab}) \cdot b$ gilt.

Da die Familie der kontextsensitiven Sprachen alle endlichen Mengen enthält und unter Vereinung abgeschlossen ist, folgt der erste Teil der Behauptung ET0L $\subseteq$ CS.

Die Echtheit der Inklusion folgt unmittelbar aus den Abschlußeigenschaften der beiden Sprachfamilien, da die Familie der ET0L-Sprachen eine AFL ist, wohingegen die Familie der CS-Sprachen *keine* AFL ist, da sie beispielsweise nicht unter Homomorphismen abgeschlossen ist. □

Abschließend noch zu deterministischen Systemen. Es zeigt sich, daß Determinismus eine echte Einschränkung für E[T]0L Sprachen darstellt.

Theorem 3.11 *ED0L* $\subset$ *E0L.*

Beweis. Die endliche Sprache $L = \{a, a^2\}$ wird von dem E0L-System $G = (\{S, a, F\}, \sigma, S, \{a\})$ mit $\sigma(S) = \{a, a^2\}$ und $\sigma(a) = \sigma(F) = \{F\}$ erzeugt, ist aber keine ED0L-Sprache.

Wie führen einen Widerspruchsbeweis. Annahme es gäbe ein ED0L-System mit Axiom ω das L erzeugt, dann muß

$$\omega \Rightarrow^* a \Rightarrow^* a^2 \quad \text{oder} \quad \omega \Rightarrow^* a^2 \Rightarrow^* a$$

gelten.

Aufgrund der D-Eigenschaft muß dann im ersten Fall $a^2 \Rightarrow^* a^4$ gelten. Dies ist ein Widerspruch, da $a^4 \notin L$. Im zweiten Fall argumentieren wir: Wenn $a^2 \Rightarrow^* a$, dann existiert ein $n \geq 1$ mit $a \Rightarrow^n a$ und $a \Rightarrow^n \lambda$, da sich a auf andere Weise nicht erzeugen läßt. Dies widerspricht aber der D-Eigenschaft, also ist L *keine* ED0L-Sprache. □

Theorem 3.12 *ED0L* $\subset$ *EDT0L.*

Beweis. Wir betrachten wieder die Sprache $L = \{a, a^2\}$, die von dem EDT0L-Sytem $G = (\{S, a, F\}, H, S, \{a\})$ mit $H = \{h_1, h_2\}$ und $h_1(S) = a, h_2(S) = a^2$ sowie $h_i(a) = h_i(F) = F$, für $i = 1, 2$, erzeugt wird. Da L laut dem vorhergehenden Theorem keine ED0L-Sprache ist, folgt die Behauptung. □

Die noch fehlende echte Inklusion von EDT0L in ET0L folgt unmittelbar aus den schon bekannten Abschlußeigenschaften der Sprachfamilien, da ET0L eine volle AFL ist, und EDT0L *nicht* unter inversen Homomorphismen abgeschlossen ist.

Theorem 3.13 *EDT0L* $\subset$ *ET0L.* □

OK ←

3.6 Zum festen Wortproblem von L-Systemen mit Hilfssymbolen

Datum: 12. Juni

In diesem Unterabschnitt beschäftigen wir uns mit dem festen Wortproblem für L-Systeme. Ganz allgemein ist das feste Wortproblem zu einem Spracherzeugungs- oder einem Spracherkennungsmechanismus wie folgt definiert:

Fest gegeben: Grammatik G (oder Maschine M).

Eingabe: Wort w.

Ausgabe: Ist $w \in L(G)$ ($w \in L(M)$)?

Man beachte, daß diese Art der Problemstellung häufig in der Praxis im Compilerbau auftritt. Fest gegeben ist hierbei die Beschreibung der zugrundeliegenden Programmiersprache, Eingabe ist das geschriebene Programm und Ausgabe ist eigentlich der erzeugte Code, für unseren Fall reicht aber ob das Programm gemäß der festen gegebenen Beschreibung korrekt ist oder nicht.

Um von Komplexität sprechen zu können führen wir noch ein paar Begriffe aus der Komplexitätstheorie ein. Zuerst zu dem Klassen, die wir benötigen:

$$AC^0 \subset NC^1 \subseteq DSpace(\log n) \subseteq NSpace(\log n) \subseteq P \subseteq NP,$$

wobei AC^0 die Klasse von Sprachen ist, die durch alternierende Turingmaschinen mit konstant beschränkter Alternierung und logarithmisch Zeitbeschränkung, NC^1 die Klasse von Sprachen, die durch alternierende Turingmaschinen mit logarithmischem Zeitbedarf, $DSpace(\log n)$ die Klasse von Sprachen, die durch deterministische Turingmaschinen mit logarithmischem Platzbedarf, $NSpace(\log n)$ die Klasse von Sprachen, die durch nichtdeterministische Turingmaschinen mit logarithmischer Plaztbeschränkung, P die Klasse von von Sprachen, die durch deterministische Turingmaschinen mit polynomieller Zeitbeschränkung und NP die Klasse von von Sprachen, die durch nichtdeterministische Turingmaschinen mit polynomieller Zeitbeschränkung akzeptiert werden.

Ein Problem oder Sprache L heißt vollständig für die Klasse A, wenn (1) $L \in A$ und (2) alle Probleme in A lassen sich auf L reduzieren. Wenn nichts anderes gesagt wird, verwenden wir deterministisch logarithmisch platzbeschränkte many-one Reduktionen.

Nun zum eigentlichen Inhalt des Kapitels. Bevor wir die L-Systeme in der Reihenfolge ihrer Erzeugungsmächtigkeit behandeln, zuerst noch zwei Vorbemerkungen: (1) Die festen Wortprobleme für ED0L-, EDT0L-, E0L- und ET0L-Sprachen sind entscheidbar, da ET0L $\subset$ CS gilt. (2) Da wir das feste Wortproblem betrachten können wir uns bis auf den ED0L-Fall immer auf propagierende Systeme zurückziehen. Dies impliziert, daß die Ableitunglänge bei einem gegebenen Wort w höchstens polynomiell in $|w|$ sein kann. Hieraus ergibt sich unmittelbar, daß die Komplexität im höchsten Fall NP sein kann.

Für das feste Wortproblem für ED0L-Sprache geben wir keinen Algorithmus an, da dieser sehr stark auf Eigenschaften dieser Systeme zugreift, die den Rahmen dieser Vorlesung bei weitem sprengen würde. Wir erwähnen das Ergebnis nur, und verweisen auf eine Arbeit von Damm, Holzer, Lange und Rossmanith [1, 2].

Theorem 3.14 *Das feste Wortproblem für ED0L-Sprachen ist in AC^0.* □

Bemerkenswert an diesem Resultat ist, daß das feste Wortproblem für ED0L-Sprachen vom komplexitätstheoretischen Standpunkt echt leichter wie das feste Wortproblem für reguläre Sprachen ist, welches NC^1-vollständig bzgl. einfacher Reduktionen ist.

Den ersten Algorithmus, den wir im Zusammenhang mit dem festen Wortporblem kennenlernen werden, ist der für EDT0L-Sprachen. Hierfür benötigen wir noch einige Bezeichnungsweisen.

Sei $G = (\Sigma, H, a_1, \Delta)$ eine festes EDT0L-System mit $\Sigma = \{a_1, a_2, \ldots, a_p\}$, $w \in \Sigma^*$ ein beliebiges Wort, von dem man feststellen will, ob $w \in L(G)$ gilt, und $\#$ ein Zeichen, welches in Σ nicht vorkommt.

Definiton 3.10 *Ein* Teilwortvektor *(von Teilworten von w auf Symbole aus Σ) ist jedes p-Tupel $(w_1, \ldots, w_p)$ so, daß jedes w_i entweder gleich $\#$ oder gleich einem Teilwort von w ist. Anstelle von $(w_1, \ldots, w_p)$ schreiben wir auch $\bar{w}$.*

Eine Teilwortvektor $\bar{w}$ ist konsistent, wenn es eine Tafelsequence $h_1, \ldots, h_m \in H$ gibt so, daß für jedes $i = 1, \ldots, p$ entweder $w_i = \#$ oder $a_i \Rightarrow_{h_1} \cdots \Rightarrow_{h_m} w_i$ gilt.

Jedes Teilwort von w wird durch die Anfangs- und Endposition in w eindeutig beschrieben. Daher kann jedes Teilwort von w durch logarithmisch viele Bits kodiert werden. Da p eine Konstante ist, kann daher jeder Teilwortvektor $\bar{w}$ durch $O(\log n)$ Bits gespeichert werden.

Definiton 3.11 *Seien $\bar{v}$ und $\bar{w}$ Teilwortvektoren. Wir schreiben $\bar{v} \to \bar{w}$ genau dann, wenn es eine Tafel $h \in H$ gibt so, daß für alle $i = 1, \ldots, p$ gilt, wenn $w_i \neq \#$, dann ist $w_i = v_{i_1} v_{i_2} \ldots v_{i_k}$, wobei $h(a_i) = a_{i_1} a_{i_2} \ldots a_{i_k}$.*

Folgender Sachverhalt ist unmittelbar einsichtig: Wenn $\bar{v}$ und $\bar{w}$ zwei Teilwortvektoren sind, $\bar{v}$ konsistent ist und $\bar{v} \to \bar{w}$ gilt, dann ist auch $\bar{w}$ konsistent.

Um den Algorithmus, für EDT0L-Sprachen angeben zu können, brauchen wir noch folgendes Lemma, welches wir nicht beweisen werden.

Lemma 3.5 *Seien $\alpha, \beta \in \Sigma^*$ so, daß $\alpha \Rightarrow_h \beta \Rightarrow_{h_1} \cdots \Rightarrow_{h_m} v$ und sei $\bar{v}$ eine konsistenter Teilwortvektor, wobei für jedes i, $a_i \Rightarrow_{h_1} \cdots \Rightarrow_{h_m} v_i$, wenn $\beta \in \Sigma^* a_i \Sigma^*$, und $v_i = \#$ sonst. Dann gibt es einen konsistenten Teilwortvektor $\bar{w}$ mit $\bar{v} \to \bar{w}$ so, daß für jedes i, $a_i \Rightarrow_h \cdots \Rightarrow_{h_m} w_i$, wenn $\alpha \in \Sigma^* a_i \Sigma^*$, und $w_i = \#$ sonst gilt.* □

Theorem 3.15 *Für jedes EDT0L-System G gilt $L(G) \in NSpace(\log n)$.*

Beweis. Mit Hilfe von Teilwortvektoren läßt sich das feste Wortproblem nun umformulieren: Ein Wort w ist in $L(G)$ genau dann, wenn es eine Sequenz von Teilwortvektoren $\bar{w}^1 \to \bar{w}^2 \to \cdots \to \bar{w}^k$ gibt mit $\bar{w}^1 = (b_1, \ldots, b_p)$ wobei $b_i = a_i$ wenn $w \in \Sigma^* a_i \Sigma^*$ und $b_i = \#$ andernfalls, und $\bar{w}^k = (w, \#, \ldots, \#)$. Beachte, daß a_1 das Axiom ist.

Der Beweis stützt sich vorallem auf Lemma 3.5 und wird dem interessierten Leser überlassen. Diese Umformulierung führt nun unmittelbar zu einem nichtdeterministischen Algorithmus:

for $i := 1$ **to** p **do**
    $w_i :=$ **if** $w \in \Sigma^* a_i \Sigma^*$ **then** a_i **else** $\#$;
$\bar{w} := (w_1, \ldots, w_p)$;
while $\bar{w} \neq (w, \#, \ldots, \#)$ **do**

begin Rate Teilwortvektor $\bar{v}$;
if $\bar{w} \to \bar{v}$ **then** $\bar{w} := \bar{v}$
else halt
end;
accept.

Da zu jedem Zeitpunkt höchstens zwei Teilwortvektoren gespeichert werden müssen, ergibt sich aufgrund der vorher gemachten Bemerkung ein Platzbedarf von $O(\log n)$ Bits. □

Korollar 3.2 *Das feste Wortproblem für EDT0L-Sprachen ist NSpace*$(\log n)$*-vollständig.* □

Beweis. Um die Vollständigkeit zu zeigen genügt es aufgrund von Theorem 3.15 ein bekanntes *NSpace*$(\log n)$-vollständiges Problem auf das feste Wortproblem für EDT0L-Sprachen zu reduzieren. Hierfür wählen wir das feste Wortproblem für linear kontextfreie Sprachen.

Sei $G = (N, T, P, S)$ eine linear kontextfreie Sprache mit den Produktionen $P = \{p_1, \ldots, p_m\}$. Definiere eine Homomorphismus $g : N \cup T \to \bar{N} \cup T$ vermöge $g(a) = \bar{a}$ wenn $a \in N$ und $g(a) = a$ andernfalls.

Offensichtlich ist das EDT0L-System $G' = (\bar{N} \cup T, H, \bar{S}, T)$ mit $H = \{h_1, \ldots, h_m\}$ wobei h_i aus den Produktionen

$$\{\, g(A) \to g(w) \mid p_i = A \to w \,\} \cup \{\, g(A) \to g(A) \mid A \in T \,\}$$
$$\cup \{\, g(A) \to g(A) \mid A \in N \text{ und } A \text{ ist } \textit{nicht} \text{ linke Seite von } p_i \,\},$$

besteht, äquivalent zu G.

Das heißt, daß man zu einer gegebene Instanz des festen Wortporblems für linear kontextfreie Sprachen eine Instanz des festen Wortproblem für EDT0L-Sprachen konstruieren kann. □

Wenden wir uns nun E0L-Sprachen zu. Zuerst geben wir einen Algorithmus zum Erkennnen von E0L-Sprachen an, der im wesentlichen eine Erweiterung des Top-Down Algorithmus für kontextfreie Sprachen darstellt.

Theorem 3.16 *Jede E0L-Sprache kann durch eine nichtdeterministische logarithmisch platzbeschränkte Turingmaschine mit Keller in polynomieller Zeit erkannt werden.*

Beweis. Sei $G = (\Sigma, \sigma, S, \Delta)$ ein EP0L-System und M ein nichtdeterministischer logarithmisch platzbeschränkte Turingmaschine mit Keller, welche folgende Instruktionen bei Eingabe w der Länge n durchführt:

1. Maschine M rät sich nichtdeterministisch eine Zahl k und schreibt sie auf das Arbeitsband (k ist die Länge jedes Pfades von der Wurzel zu den Blättern im Ableitungsbaum von w).

2. M schreibt Kellerbodensymbol # und das Paar $\langle S, 0\rangle$, geeignet durch die Kellersymbole kodiert, auf den Keller.

3. Für $0 \leq i < k$, wenn $\langle A, i\rangle$ das oberste Element des Kellers ist, dann ersetzt M es durch das Wort $\langle B_1, i+1\rangle \ldots, \langle B_m, i+1\rangle$, wenn $B_1 \ldots B_m$ Element von $\sigma(A)$ ist. Hierbei ist der Keller so zu organisieren, daß als nächstes das Paar $\langle B_1, i+1\rangle$ ausgelesen wird.

4. Wenn (A, k) das oberste Element des Kellers ist, dann überprüft M ob A das aktuelle Eingabezeichen ist oder nicht und ob $A \in \Delta$ gilt oder nicht. Wenn A nicht das aktuelle Eingabezeichen ist, oder $A \notin \Delta$, dann hält M and lehnt die Eingabe ab. Andernfalls löscht M das Tupel (A, k), bewegt den Eingabekopf nach rechts und geht zu Schritt 5 über.

5. Wenn das oberste Kellersymbol nicht # ist, dann wiederholt M die Schritte 3 bis 5. Wenn das Kellerbodensymbol erreicht ist, und die gesamte Eingabe *nicht* gelesen ist, dann hält M und lehnt ab. Andernfalls hält M und akzeptiert die Eingabe w.

In der Einführung dieses Abschnitts habe wir argumentiert, daß die Ableitunglänge polynomiell beschränkt ist. Daher kann die Zahl k auf logarithmischen Platz notiert werden. Weiterhin ist damit klar, daß der angegebene Algorithmus in polynomieller Zeit zu einem Ergebnis kommt. Dies zeigt die Behauptung. □

Da der Rahmen dieser Veranstaltung sehr begrenzt ist, bleibt uns nicht die Zeit, zu zeigen, daß das soeben eingeführte Automatenmodell äquivalent zum logspace many-one Abschluß der kontext-freien Sprachen ist [11]. Da darüberhinaus auch CFL $\subseteq$ E0L nach Theorem 3.8 gilt, folgt damit.

Korollar 3.3 *Das feste Wortproblem für E0L-Sprachen ist äquivalent zum Wortproblem für kontextfreie Sprachen.* □

Abschließend zeigen wir noch, daß das feste Wortproblem für ET0L-Sprachen *NP*-vollständig ist.

Theorem 3.17 *Das feste Wortproblem für ET0L-Sprachen ist NP-vollständig.*

Beweis. In der Einleitung dieses Abschnitts haben wir argumentiert, daß die Länge einer Ableitung höchstens polynomiell beschränkt ist. Diese Tatsache läßt sich nutzen um zu zeigen, daß für jedes ET0L-System G gilt: $L(G) \in NP$.

Um die untere Schranke zu beweisen, verwenden wir das *NP*-vollständige Erfüllbarkeitsproblem für Aussagenlogische Formeln (SAT) in konjunktiver Normalform, und zeigen, daß eine geeignete Kodierung von SAT durch ein ET0L-System erzeugbar ist.

Eine Formel $f = C_1 \wedge \ldots \wedge C_p$ mit m Variablen aus SAT besteht aus Klauseln C_i, welche Disjunktionen von Variablen x_j oder Negationen von Variablen $\neg x_j$ sind. Anstelle eine Klausel C in gewöhnlicher Art und Weise hinzuschreiben, kodieren wir diese durch ein Wort w_C der Länge m über dem Zeichenvorrat $\{0, -, +\}$ wie folgt: An Position i, mit $1 \leq i \leq m$ von w_C steht eine 0, falls Variable x_i in C

nicht vorkommt, ein $-$ falls $\neg x_i$ in C vorkommt und $+$ falls Variable x_i in C vorkommt.

Zum Beispiel wir die Klausel $x_1 \vee \neg x_3 \vee x_4$ durch $+0-+$ kodiert.

In natürlicher Weise erweitern wir die Kodierung von Klauseln auf Formeln $f = C_1 \wedge \ldots \wedge C_p$ in dem wir die Kodierung der einzlnen Klauseln durch # getrennt aneinanderhängen. Für f ergibt das dann das Wort der Form $w_{C_1}\# \ldots w_{C_p}\#$ wobei w_{C_i} die Kodierung der $i$ten Klausel C_i ist.

Die Idee ist es nun ein ET0L-System $G = (\Sigma, H, S, \{0, -, +, \#\})$ zu konstruieren, das alle Formeln in kodierter Form erzeugt, die erfüllbar sind. Setze

$$\Sigma = \{S, D, T, F, [T, 0], [T, 1], [F, 0], [F, 1], 0, -, +, \#\}.$$

Die nachfolgende Tabelle soll erklären für welchen Zweck die Symbole aus Σ verwendet werden:

Symbol:	Findet Verwendung als ...
S	Startsymbol
D	Fehlersymbol (im bisherigen Teil des Skrptes war das immer F!)
T	der Teil der bisher erzeugte Klausel ist *wahr* (der Rest der Klausel ist unabhängig von der weiteren Belegung)
F	der Teil der bisher erzeugte Klausel ist *falsch*
$[T, 0]$	der Teil der bisher erzeugte Klausel ist *wahr* und die zu diesem Zeitpunkt betrachtete Variable wird mit *falsch* belegt
$[T, 1]$	der Teil der bisher erzeugte Klausel ist *wahr* und die zu diesem Zeitpunkt betrachtete Variable wird mit *wahr* belegt
$[F, 0]$	der Teil der bisher erzeugte Klausel ist *falsch* und die zu diesem Zeitpunkt betrachtete Variable wird mit *falsch* belegt
$[F, 1]$	der Teil der bisher erzeugte Klausel ist *falsch* und die zu diesem Zeitpunkt betrachtete Variable wird mit *wahr* belegt
$0, -, +, \#$	terminale Symbole

Die einzelnen Tafeln des Systems sind nachfolgend beschrieben.

Um die Ableitung zu beginnen, werden Produktionen aus der ersten Tafel

$$h_1 = \{S \to FS, S \to F, F \to F\} \cup \{0 \to D, - \to D, + \to D, \# \to D\} \cup \{D \to D\}$$

verwendet. Nach mehrmaligem Anwenden dieser Tafeln erhalten wir eine Ableitung der Form $S \Rightarrow F^p$.

Nun rät sich das ET0L-System wie die nächste Variable der Formel zu belegen ist. Hierzu finden entweder die Regeln aus zweiten oder dritten Tafeln Anwendung:

$$h_2 = \{T \to [T,0], F \to [F,0]\} \\ \cup \{0 \to 0, - \to -, + \to +, \# \to \#\} \cup \{S \to D, D \to D\}$$

$$h_3 = \{T \to [T,1], F \to [F,1]\} \\ \cup \{0 \to 0, - \to -, + \to +, \# \to \#\} \cup \{S \to D, D \to D\}$$

Nachdem die Belegung der Variablen in den geklammerten Zeichen gespeichert ist, kann man nun das nächste Bit der Kodierung der Klauseln erzeugen. Dies geschieht mit

$$h_4 = h_{41} \cup h_{42} \cup h_{43} \\ \cup \{0 \to 0, - \to -, + \to +, \# \to \#\} \cup \{S \to D, D \to D\},$$

wobei

$$\begin{aligned} h_{41} &= \{[T,0] \to 0T, [F,0] \to 0F\} \cup \{[T,1] \to 0T, [F,1] \to 0F\}, \\ h_{42} &= \{[T,0] \to -T, [F,0] \to -T\} \cup \{[T,1] \to -T, [F,1] \to -F\} \quad \text{und} \\ h_{43} &= \{[T,0] \to +T, [F,0] \to +F\} \cup \{[T,1] \to +T, [F,1] \to +T\}. \end{aligned}$$

Zum Beispiel wird durch Anwendung einer Regel aus h_{41} simuliert, daß diese Variable in der Klausel *nicht* vorkommt, und mit einer Regel aus h_{43}, daß die Variable positiv vorkommt.

Die Terminierung geschieht schließlich mit

$$h_5 = \{T \to \#\} \\ \cup \{0 \to 0, - \to -, + \to +, \# \to \#\} \cup \{F \to D, S \to D, D \to D\}.$$

Am Beispiel der Formel $f = (x_1 \vee \neg x_3 \vee x_4) \wedge x_2$ soll die Arbeitsweise des ET0L-System verdeutlich werden. Die Kodierung der Formel f lautet $+0-+\#0+00\#$. Eine mögliche Ableitung sieht wie folgt aus:

$$\begin{aligned} S &\Rightarrow_{h_1} FS \Rightarrow_{h_1} FF \\ &\Rightarrow_{h_2} [F,0][F,0] \Rightarrow_{h_4} +F0F \\ &\Rightarrow_{h_3} +[F,1]0[F,1] \Rightarrow_{h_4} +0F0+T \\ &\Rightarrow_{h_2} +0[F,0]0+[T,0] \Rightarrow_{h_4} +0-T0+0T \\ &\Rightarrow_{h_2} +0-[T,0]0+0[T,0] \Rightarrow_{h_4} +0-+T0+00T \\ &\Rightarrow_{h_5} +0-+\#0+00\#. \end{aligned}$$

Aus der Anwendung der Tafeln h_2 und h_3 läßt sich dann eine erfüllenden Belegung für die Formel rekonstruieren. In diesem Fall wurde x_1 mit falsch, x_2 mit wahr, x_3 mit falsch und x_4 ebenfalls mit falsch belegt.

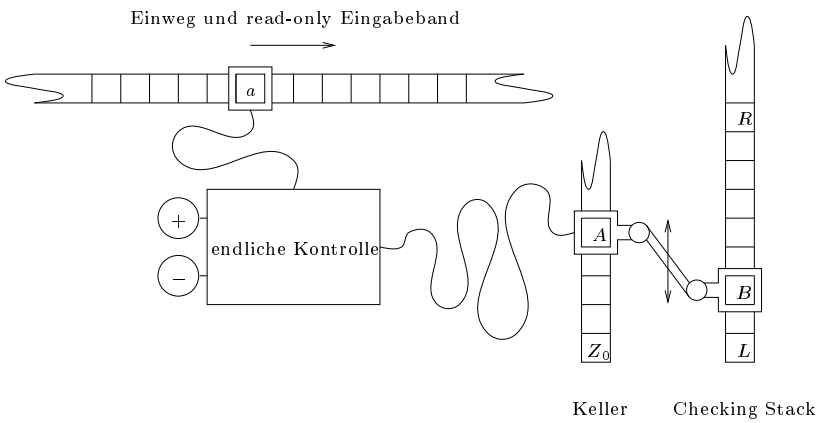
Wie man sich nun leicht überzeugt, erzeugt G nur erfüllbare aussagenlogische Formeln. Damit wäre die Behauptung bewiesen. □ OK ←

3.7 Automatenmodelle für L-Systeme mit Hilfssymbolen

Datum: 19. Juni

In diesem Unterabschnitt betrachten wir Automatenmodelle für L-Systeme mit Hilfssymbolen. Wir betrachten ein top-down arbeitendes Automatenmodell für die Familie der ET0L-Sprachen, welches eine Erweiterung des klassischen Kellerautomaten darstellt. Da die Beweise bei diesen Automaten sehr technisch und länglich sind wird im nachfolgenden nur die Beweisidee vorgestellt.

Ein *Checking Stack Kellerautomat* (CS-PD) ist eine Machine mit der Struktur wie im folgenden Bild.



Der externe Speicher besteht aus einem gewöhlichen Keller in Kombination mit einem sogenannten Checking Stack. Bevor eine Berechnung auf Eingabe w beginnt, wird der Checking Stack mit einem beliebigen Wort $y \in \Theta^*$ gefüllt, welches eingeschlossen wird durch den linken- und rechten Endmarker L und R, wobei Θ das Checking Stack Alphabet ist. In den Keller wird das Kellerbodensymbol Z_0 geschrieben. Die Bewegung des Schreib-Lesekopfs des Kellers und des Lesekopfs des Checking Stacks sind folgendermaßen verbunden: Jede push und pop Bewegung des Kellers impliziert eine Bewegung der gleichen Art auf dem Checking Stack.

Am Anfang der Berechnung stehen die Köpfe des externen Speichers auf ihrer linkest möglichen Position. D.h., der Kopf des Kellers liest das Kellerbodensymbol, der Kopf des Checking Stacks liest das Symbol L, der Eingabekopf liest das erste Symbol der Eingabe w, und die endliche Kontrolle befindet sich im Startzustand. Ein Schritt des CS-PD Automaten M sieht wie folgt aus: M liest das aktuelle Symbol auf dem Keller, das aktuelle Symbol auf dem Checking Stack und das aktuelle Eingabesymbol bzw. λ und entscheidet sich aufgrund dieser Information und des aktuellen Zustands, in einen neuen Zustand zu wechseln, ein Symbol auf den Keller zu pushen, zu poppen, oder zu überschreiben und den Eingabekopf entweder ein Symbol nach rechts zu bewegen—falls kein λ gelesen wurde—oder den Eingabekopf auf der Position stehen zu lassen.

Die Transitionsfunktion δ eines CS-PD sieht damit wie folgt aus

$$\delta : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \times (\Theta \cup \{L, R\}) \to 2^{Q \times I},$$

wobei Q die endliche Menge von Zuständen, Σ das Eingabealphabet, Γ das Kelleralphabet mit dem Kellerbodensymbol Z_0, Θ das Checking Stack Alphabet, und I die Menge der Instruktionen ist. Eine Instruktion ist entweder ein pop, push oder overwrite. Die Transitionfunktion δ beschreibt einen Schritt des Automaten wie oben beschrieben.

Ein CS-PD M akzeptiert ein Wort w, wenn es eine Folge von Schritten von M, mit einem geratenen Inhalt des Checking Stacks gibt so, daß das gesamte Wort w gelesen wurde und der Keller leer ist (eine äquivalente Definition wäre die Akzeptanz durch Endzustand). Die Familie der Sprachen die durch CS-PD Automaten akzeptiert wird, bezeichnen wir mit CS-PD.

Van Leeuwen [14] zeigte, daß die Familie der CS-PD-Sprachen äquivalent zur Familie der ET0L-Sprachen ist.

Theorem 3.18 *CS-PD = ET0L.*

Intuitiv gesprochen wird ein ET0L-System G durch einen CS-PD Automaten M folgendermaßen simuliert: M rät sich die Abfolge der Tafeln von G und schreibt sie in den Checking Stack. Anschließend wird im Keller eine Linksableitung in top-down Weise für eine gültige Ableitung (richtige Anwendung der Tafeln) konstruiert. Die andere Richtung der Simulation, nämlich ein CS-PD Automat wird durch ein ET0L-System simuliert, kann durch eine Adaption der klassischen Simulation von Kellerautomat durch kontextfreie Grammatik gezeigt werden.

Einige natürliche Einschränkungen von CS-PD Automaten erlauben es, Unterklassen von ET0L zu charakterisieren. Für eine deterministische Einschränkung ist eine etwas andere Sichtweise von CS-PD Automaten von Vorteil.

Ein CS-PD Automat kann wie folgt als ein Transduktor aufgefaßt werden: Der Inhalt des Checking Stacks kann als Eingabe gesehen werden, der Keller wir zum externen Speicher und das Eingabeband des CS-PD wird zum Ausgabeband des Transduktors.

Auf diese Weise kann ein CS-PD Automat als eine eingeschränkte Variante eines Kellertransduktors gesehen werden. Das Lesen eines Zeichens a im CS-PD Automaten entspricht nun der Ausgabe von a im Transduktor (falls der Automat λ gelesen hat, dann wird auch λ ausgegeben). Die Transitionsfunktion ist dann eine Abbildung von $Q \times \Gamma \times (\Theta \cup \{L, R\})$ nach $2^{Q \times I \times (\Sigma \cup \{\lambda\})}$. Auf diese Weise sprechen wir von einem CS-PD Traksduktor.

Die Ausgabesprache eines CS-PD Transduktors M ist die Menge aller Wörter über Σ^*, die bei Eingabe von Worten $y \in \Theta^*$ während einer korrekten Berechung M bei Erreichen des leeren Kellers ausgegeben werden. Man kann sich leicht überzeugen, daß die Familie der CS-PD Transduktor Sprachen identisch zur Familie der CS-PD Sprachen ist.

Mit dieser Sichtweise können wir nun Determinismus wie folgt definieren. Ein CS-PD Transduktor ist *deterministisch* genau dann, wenn für jedes (q, Z, A) in $Q \times \Gamma \times (\Theta \cup \{L, R\})$ die Bedingung $|\delta(q, Z, A)| = 1$ gilt. Die Familie der durch deterministische CS-PD Transduktoren erzeugten Sprachen wird mit DCS-PD

bezeichnet. Mit Hilfe dieser Art der Einschränkung von CS-PD Transduktoren gelang des Engelfiet, Schmidt und van Leeuwen, die Familie der EDT0L-Sprachen zu kennzeichnen.

Theorem 3.19 *DCS-PD = EDT0L.*

Aufgrund der intuitiven Beschriebung der Beweisidee CS-PD = ET0L könnte man nun glauben, daß CS-PD Automaten mit einelementigem oder unärem Checking Stackalphabet Θ genau die Familie der E0L-Sprachen charakterisiert. Dem ist leider nicht, so wie man im nachfolgenden Theorem sieht.

Theorem 3.20 *UCS-PD* $= \mathcal{H}^1(REG)$.

Hierbei bezeichnet UCS-PD die Familie der CS-PD-Sprachen die durch CS-PD Automaten mit unärem Checking Stack Alphabet erkannt werden, und $\mathcal{H}^1$(REG) bezeichnet die Familie der Sprachen die durch sogenannte reguläre 1-beschränkte Iterationsgrammatiken erzeugt werden (dies sind nichts anderes als erweiterte E0L-Systeme, wobei die rechten Seiten der Regelmenge durch reguläre Sprachen und nicht durch endliche Mengen beschrieben werden). Offensichtlich gilt $\mathcal{H}$(FIN) = E0L, wobei FIN die Menge der endlichen Sprachen bezeichnet. Darüberhinaus kann man folgendes zeigen:

Theorem 3.21 *E0L* $\subset \mathcal{H}^1(REG) \subset$ *ET0L.*

Um die E0L-Sprachen durch CS-PD Automaten oder Transduktoren zu beschreiben, wird noch eine stärkere Einschränkung als unäres Checking Stack Alphabet benötigt. Diese Bedingung ist die sogenannte LFR-Eigenschaft. Ein CS-PD Automat M besitzt die LFR-Eingenschaft, wenn eine Konstante s existiert so, daß wenn M während einer erfolgreichen Berechnung eine Zelle c des Kellers besucht, dann besucht M diese Zelle höchstens s-mal, bevor c das erste mal gepoppt wird. Wenn wir die Familie von Sprachen die durch UCS-PD Automaten mit der LFR-Eigenschaft mit LFR-UCS-PD bezeichnen, erhalten wir (ein Beweis findet sich bei van Leeuwen [13]):

Theorem 3.22 *LFR-UCS-PD = E0L.*

Abschließend sei noch angemerkt, daß es noch ein anderes Automatenmodell für die Familie der ET0L-Sprachen und der Unterklassen EDT0L, $\mathcal{H}^1$(REG) und E0L gibt. Ähnlich wie CS-PD Automaten ist dieses Modell eine Erweiterung des klassischen Kellerautomaten, der nun aber bottom-up und nicht top-down arbeitet. Dieser Automat ist in der Literatur unter dem Namen *Pushdown array of pushdowns* (PAP) Automat bekannt. Grob gesprochen ist dies eine Kellerautomat dessen Kellersymbole wieder einzelne Keller sind. Die exakte Beschreibung des Automaten ist langwierig und wir verweisen deshalb auf die Erläuterungen von Rozenberg und Salomma [9] auf Seite 306–307.

Ähnlich wie man bei CS-PD Automaten zu UCS-PD Auomaten übergegangen ist, kann man dies auch bei PAP Automaten tun, indem man *pushdown array of counters* (PAC) betrachtet. Für PAP und PAC Automaten bzw. Sprachen findet man die Beziehung:

Theorem 3.23 *1. $PAP = ET0L$ und*

2. $PAC = \mathcal{H}^1(REG)$.

Um darüberhinaus die Familie der E0L-Sprachen zu charakterisieren, muß noch eine weitere Bedingung an PAP Automaten gestellt werden. Diese Einschränkung ist vergeichbar mit der LFR-Eigenschaft von CS-PD Automaten. Wir gehen auf diese Einschränkung nicht näher ein, und verweisen auf Rozenberg und Wood [10] und Rozenberg [8]. Wenn man PAP und PAC Automaten mit dieser Einschränkung als RPAP und RPAC Automaten bezeichnet, und die zugehörigen Sprachfamilien entsprechend bezeichnet, dann erhält man:

Theorem 3.24 *1. $RPAP = ET0L$ und*

2. $RPAC = E0L$.

OK ←

3.8 Ein anderer Spracherzeugungsmechanismus

Datum: 26. Juni

Aus der Vielzahl der Spracherzeugungsmechanismen der Gruppe **D** (siehe Tabelle in Abschnitt 3.2) haben wir bisher nur den E-Mechanismus genauer kennengelernt. In diesem Abschnitt wollen wir uns A-Mechanismus zuwenden. Wie wir sehen werden, erlaubt dieser eine Characterisierung der kontextfreien Sprachen mit Hilfe von 0L-Systemen. Im übrigen hat auch dieser Mechanismus einen biologischen Ursprung. Es steckt die Idee dahinter, das Pflanzen oder Organismen im allgemeinen nach einer Wachstumsphase in einen „Erwachsenenzustand“ übergehen. Nun aber zur formalen Definition:

Definiton 3.12 *Sei $G = (\Sigma, \sigma, \omega)$ ein 0L-System. Die „*adult*“ Sprache von G ist definiert durch*

$$L_A(G) = \{\, w \in L(G) \mid w \Rightarrow w' \textit{ impliziert } w = w' \textit{ für alle } w'. \,\}$$

Die Familie der adult 0L Sprachen wird mit A0L bezeichnet.

Zur einfacheren Bezeichnung führen wir folgende Kurzschreibweisen ein: (1) $x \Rrightarrow y$ bedeutet, daß $x \Rightarrow y$ und aus $x \Rightarrow y_1$ folgt $y = y_1$. (2) $x \overset{m}{\Rrightarrow} y$ bedeutet, daß $x \Rightarrow^m y$ und aus $x \Rightarrow^m y_1$ folgt $y = y_1$.

Damit läßt sich die Definition der adult Sprache wie folgt beschreiben:

$$L_A(G) = \{\, w \in L(G) \mid w \Rrightarrow w \,\}.$$

Um die A0L-Sprachen in Ihrer Mächtigkeit einzuschätzen, werden wir die Sprachfamilie innerhalb der Chomsky-Hierarchie einordnen. Dazu benötigen wir noch einen Begriff, der eine Art „Terminalalphabet“ beschreibt.

Definiton 3.13 *Sei $G = (\Sigma, \sigma, \omega)$ ein 0L-System. Das adult Alphabet Σ_A sei die Untermenge von Σ, die alle Zeichen enthält, die in Wörtern aus $L_A(G)$ vorkommen, d.h.*

$$\Sigma_A = \{\, a \in \Sigma \mid \exists w \in L_A(G) \text{ so, daß } |w|_a > 0 \,\}.$$

Wir wollen nun einen Algorithmus entwickeln, der uns zu einem gegebenen 0L-System das adult Alphabet der zugehörigen A0L-Sprache berechnet. Dazu benötigen wir noch folgendes Lemma.

Lemma 3.6 *Sei $G = (\Sigma, \sigma, \omega)$ ein 0L-System und $m = |\Sigma_A|$. Dann existiert für jedes $a \in \Sigma_A$ ein eindeutiges Wort $x_a \in \Sigma_A^*$, so daß*

$$a \overset{m}{\Rrightarrow} x_a \Rrightarrow x_a.$$

Beweis. Das Lemma ist trivialerweise wahr, wenn Σ_A leer ist. Sei also $|\Sigma_A| > 0$ und $a \in \Sigma_A$ beliebig. Aus der Definition von $L_A(G)$ folgt, daß in σ genau eine Produktion $a \to \alpha$ mit a auf der linken Seite existiert. Außerdem kann a höchstens einmal in α vorkommen, ansonsten könnten unbegrenzt viele weitere a's abgeleitet werden, und a könnte in keinem Wort aus $L(G)$ vorkommen, was ein Widerspruch zur Annahme ist.

Wir unterscheiden also zwei Fälle:

$|\alpha|_a = 0$: Offensichtlich ist das Lemma mit $x_a = \lambda$ wahr, wenn $a \overset{t}{\Rrightarrow} \lambda$ für ein $t \leq m$.

Weil $a \in \Sigma_A$ ist, gilt mit $a \Rightarrow^t y$ auch $a \overset{t}{\Rrightarrow} y$. Gilt $\alpha = \lambda$, so ist alles klar. Sei also $\alpha \neq \lambda$. Sei $w \in L_A(G)$ mit $|w|_a > 0$. Buchstabe a kommt nicht in α vor, also muß wegen $w \Rrightarrow w$ gelten, daß w von folgender Gestalt ist:

$$w = x_1 \alpha x_2 a x_3 \quad \text{oder} \quad w = x_1 a x_2 \alpha x_3 \tag{1}$$

wobei $x_1, x_2, x_3 \in \Sigma_A^*$.

Betrachte die Folge $\alpha_1, \alpha_2, \ldots$ von Wörtern mit $a \overset{i}{\Rrightarrow} \alpha_i$. Dann folgt aus (1), daß $|w| \geq |\alpha_1 \cdots \alpha_i|$ für alle i. Dies ist nur möglich, falls für ein t gilt, daß $\alpha_t = \lambda$, d.h. $a \overset{t}{\Rrightarrow} \lambda$. Daher kann a in keinem der Wörter α_i vorkommen. Allgemein gilt: falls b in α_i vorkommt, dann enthält das Teilwort von α_{i+1}, das von b erzeugt wurde, selbst kein b mehr. Daraus folgt, daß $t \leq m$.

$|\alpha|_a = 1$: Sei also $\alpha = \beta_1 a \gamma_1$, wobei a nicht in β_1 bzw. γ_1 vorkommt. Weil a keine beliebig langen Wörter erzeugt, muß gelten, daß $\beta_1 \gamma_1 \Rightarrow^* \lambda$. Mit der gerade bewiesenen Behauptung im Fall $|\alpha|_a = 0$ folgt dann, daß $\beta_1 \overset{m}{\Rrightarrow} \lambda$ und $\gamma_1 \overset{m}{\Rrightarrow} \lambda$. Gelte $\beta_1 \Rrightarrow \beta_1 \Rrightarrow \cdots \Rrightarrow \beta_u \Rrightarrow \lambda$ und $\gamma_1 \Rrightarrow \gamma_1 \Rrightarrow \cdots \Rrightarrow \gamma_v \Rrightarrow \lambda$. Dann ist das Lemma mit

$$x_a = \beta_u \cdots \beta_2 \beta_1 a \gamma_1 \gamma_2 \cdots \gamma_v$$

erfüllt. □

Theorem 3.25 *Es existiert ein Algorithmus, der das adult Alphabet eines gegebenen 0L-Systems berechnet.*

Beweis. Der Algorithmus funktioniert wie folgt: Gegeben ein 0L-System $G = (\Sigma, \sigma, \omega)$. Sei $n = |\Sigma|$.

1. Für jedes a aus Σ überprüfe, ob für ein x_a gilt:

   $$a \overset{n}{\Rightarrow} x_a \Rightarrow x_a.$$

   Sei Σ_1 die Vereinigung aller in den x_a vorkommenden Zeichen:

   $$\Sigma_1 = \bigcup_{\substack{a \in \Sigma \\ a \overset{n}{\Rightarrow} x_a \Rightarrow x_a}} alph(x_a).$$

   Mit Lemma 3.6 wissen wir, daß $\Sigma_A \subseteq \Sigma_1$.

2. Berechne $\Sigma_2 \subseteq \Sigma_1$ so, daß für jedes $b \in \Sigma_2$ ein Wort w in $\Sigma_1^+ \cap L(G)$ existiert, so daß in w das Zeichen b vorkommt. Σ_2 kann unter Zuhilfenahme der Menge

   $$\{\, a \mid w \in L(G) \text{ mit } |w|_a \neq 0 \,\}$$

   gefunden werden. Diese Menge ist konstruierbar.

3. Σ_3 sei die Menge aller in x vorkommenden Zeichen, wobei für x gelten soll

   $$a \overset{n}{\Rightarrow} x \Rightarrow x \quad \text{für ein} \quad a \in \Sigma_2$$

Dann gilt $\Sigma_3 = \Sigma_A$. □

Um eine einfachen Simulation von A0L-Sytemen durch CF-Systeme geben zu können, benötigen wir noch eine Art „Normalform“.

Lemma 3.7 *Sei H ein 0L-System. Dann kann ein 0L-System $G = (\Sigma, \sigma, S)$ konstruiert werden, so daß $L_A(G) = L_A(H)$ gilt, und für jedes a aus dem adult Alphabet von G nur die Produktion $a \to a$ in σ exisitiert.*

Beweis. Wir geben nur den Umwandlungsalgorithmus an. Ohne Einschränkungen nehmen wir an, daß das Axiom von H das Zeichen S ist, das nicht im adult Alphabet Σ_A von H ist.

Definiere den Homomorphismus

$$h(a) = \begin{cases} x_a & \text{so, daß } a \overset{m}{\Rightarrow}_H x_a \Rightarrow_H x_a \text{ für } a \in \Sigma_A \\ a & \text{sonst} \end{cases}.$$

Sei weiter

$$\sigma = \{\, a \to h(\alpha) \mid a \to \alpha \text{ ist Produktion in } H \text{ und } a \notin \Sigma_A \,\} \cup \{\, a \to a \,|\, a \in \Sigma_A \,\}.$$

Dann gilt $L_A(G) = L_A(H)$. Insbesondere sind auch die adult Alphabete gleich. □

Nun kommen wir zu Hauptergebnis dieses Abschnitts:

Theorem 3.26 *A0L = CF.*

Beweis. Wieder geben wir nur die Simulationen an.

Sei ein 0L-System $H = (\Sigma, \sigma, S)$ gegeben, wobei S nicht aus dem adult Alphabet Σ_A stammt, das mit dem Algorithmus aus Theorem 3.25 berechnet werden kann. Wegen Lemma 3.7 können wir annehmen, daß H für jedes $a \in \Sigma_A$ nur die Produktionen $a \to a$ enthält. Wir definieren die kontextfreie Grammatik

$$G = (\Sigma \setminus \Sigma_A, \Sigma_A, \sigma \setminus \{ a \to a \mid a \in \Sigma_A \}, S).$$

Dann gilt $L(G) = L_A(H)$.

Sei umgekehrt eine kontextfreie Grammatik $G = (N, T, P, S)$ gegeben. Wir können annehmen, daß G reduziert ist (jedes Nonterminal ist von S aus erreichbar und kann terminiert werden). Außerdem gehen wir davon aus, daß S nicht auf rechten Regelseiten vorkommt, P keine Kettenproduktionen enthält und höchstens S λ erzeugen kann. Definiere das 0L-System

$$H = (N \cup T, P \cup \{ a \to a \mid a \in T \}, S).$$

Dann gilt $L_A(H) = L(G)$. □

Im Gegensatz dazu ist der A-Mechanismus angwendet auf ein D0L-System $G = (\Sigma, h, \omega)$ eine sehr starke Einschränkung, denn die zugehörige AD0L-Sprache $L_A(G)$ ist *einelementig*! Dies ist offensichtlich, da wir eine Ableitungskette der Form

$$\omega \Rightarrow h(\omega) \Rightarrow h(h(\omega)) \Rightarrow \cdots \Rightarrow h^i(\omega) \Rightarrow h^{i+1}(\omega) \Rightarrow \cdots$$

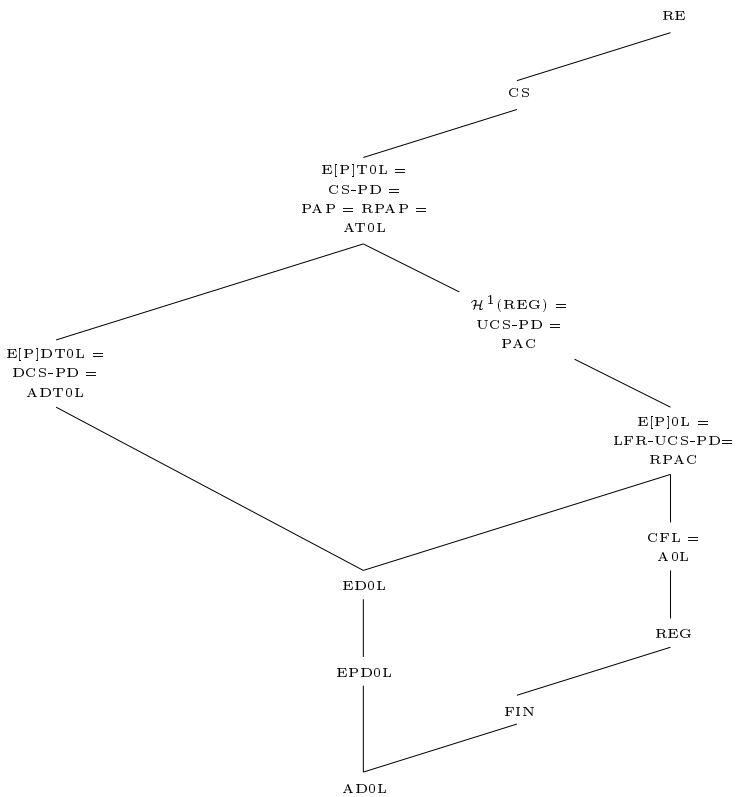
haben, und es dann ein i geben muß mit $h^i(\omega) = h^{i+1}(\omega)$, was sofort $h^i(\omega) = h^{i+j}(\omega)$ für jedes $j \geq 0$ impliziert. Da jede einelementige Sprache auch eine [E]D0L-Sprache ist, aber nicht umgekehrt, erhalten wir:

Theorem 3.27 *AD0L ⊂ ED0L.* □

Abschließend sei noch daraufhingewiesen, daß für [D]T0L-Sprachen der A- und E-Mechanismus equivalent ist. Hier verweisen wir auf Lange [5] und Vitányi und Walker [17].

Theorem 3.28 *A[D]T0L = E[D]T0L.* □

Die in Abschnitt 3 erzielten Ergebnisse zusammen mit den bekannte Ergebnissen zur Chomsky-Hierarchie fassen wir im nachfolgenden Inklusionsdiagramm zusammen. Eine Sprachfamilie A ist Obermenge von B falls A mit B durch eine Linie verbunden ist, und A über B steht. Alle dargestellten Inklusion sind *echt*. Sprachklassen die nicht durch eine Linie verbunden sind, sind nicht notwendiger Weise unvergleichbar.



OK ←

4 Kontextsensitive Lindenmayersysteme

In den bisherigen Abschnitten habe wir uns auf kontextfreie L-Systeme konzentriert. Ähnlich wie bei Chomsky-Sprachen kann man auch bei L-Systemen kontextsensitive Ersetzung definieren. In den nächsten zwei Unterabschnitten wenden wir uns zuerst den Grundlagen von kontextsensitiven L-Systemen zu, um anschließend ihre generative Mächtigkeit genauer zu studieren.

4.1 Grundlagen zu IL-Systemen

Datum: 3. Juli

Wir beginnen mit der Definition von IL-Systemen:

Definiton 4.1 *Seien $k, \ell \in \mathbb{N}_0$. Ein* $\langle k, \ell \rangle$-System *ist ein 4-Tupel $G = (\Sigma, \sigma, g, \omega)$, wobei Σ ein endliches Alphabet, g ein beliebiges Zeichen mit $g \notin \Sigma$ (der Marker), $\omega \in \Sigma^*$ das Axiom, und σ eine endliche nichtleere Menge von Produktionen mit $\sigma \subseteq (\Sigma \cup \{g\})^k \times \Sigma \times (\Sigma \cup \{g\})^\ell \times \Sigma^*$ ist.*

Es soll gelten:

1. *Falls $(w_1, a, w_3, w_4) \in \sigma$, dann*

(a) *Falls* $w_1 = u_1 g v_1$ *für* $u_1, v_1 \in (\Sigma \cup \{g\})^*$, *dann ist* $u_1 \in \{g\}^*$.

(b) *Falls* $w_3 = u_3 g v_3$ *für* $u_3, v_3 \in (\Sigma \cup \{g\})^*$, *dann ist* $v_3 \in \{g\}^*$.

2. *Für alle* $(w_1, a, w_3) \in (\Sigma \cup \{g\})^k \times \Sigma \times (\Sigma \cup \{g\})^\ell$ *gibt es ein* w_4 *in* Σ^*, *so daß* $(w_1, a, w_3, w_4) \in \sigma$, *falls* w_1 *und* w_3 *die oberen beiden Bedingungen erfüllen.*

Die $\langle k, \ell \rangle$*-Systeme werden auch als* IL-Systeme *bezeichnet.*

Anstelle von $(w_1, a, w_3, w_4) \in \sigma$ schreiben wir auch $(w_1, a, w_3) \to_\sigma w_4$ oder wenn keine Mißverständnisse auftreten auch $(w_1, a, w_3) \to w_4$. Manchmal bezeichnen wir w_1 als den Linkskontext und w_3 als den Rechtskontext von a.

Bevor wir zur Definition der Ableitungsrelation $\Rightarrow$ für IL-Systeme kommen, benötigen wir folgende vereinfachende Schreibweisen: Für ein Wort w bezeichnen wir mit $\mathrm{Suf}_n(w)$ den Suffix der Länge n von w, mit mit $\mathrm{Pref}_n(w)$ bezeichnen wir den Präfix der Länge n.

Definiton 4.2 *Seien* $G = (\Sigma, \sigma, g, \omega)$ *ein* $\langle k, \ell \rangle$*-System. Wir definieren: Wort* $w_1 = a_1 \dots a_m \in \Sigma^*$ *leitet* w_2 *(in* G*) direkt ab (in Zeichen* $w_1 \Rightarrow_G w_2$*) falls*

$$\begin{array}{rcl} (g^k, a_1, \mathit{Pref}_\ell(a_2 \dots a_m g^\ell)) & \to & \alpha_1, \\ (\mathit{Suf}_k(g^k a_1), a_2, \mathit{Pref}_\ell(a_3 \dots a_m g^\ell)) & \to & \alpha_2, \\ & \vdots & \\ (\mathit{Suf}_k(g^k a_1 \dots a_{m-1}), a_m, g^\ell) & \to & \alpha_m \end{array}$$

und $w_2 = \alpha_1 \dots \alpha_m$.

In Analogie zu Definition 3.5 lassen sich auch die Ableitungsrelation: leitet in n *Schritten* ($\Rightarrow_\sigma^n$)*, leitet echt ab* ($\Rightarrow_\sigma^+$) *und leitet ab* ($\Rightarrow_\sigma^*$) *definieren.*

Falls keine Verwechslungen möglich sind, wird auch $\Rightarrow$, $\Rightarrow^n$, $\Rightarrow^+$ und $\Rightarrow^*$ verwendet. Die von einem $\langle k, \ell \rangle$-System G erzeugte Sprache $L(G)$ ist definiert durch

$$L(G) = \{\, v \in \Sigma^* \mid \omega \Rightarrow^* v \,\}.$$

Die aus dem kontextfreien Fall bereits bekannten Erweiterungen und Einschränkung lassen sich nun auch auf IL-Systeme übertragen. Im ersten Schritt führen wir Tafeln ein:

Definiton 4.3 *Seien* $k, \ell \in \mathbb{N}_0$. *Ein* $\langle k, \ell \rangle$*-System mit Tafeln ist ein 4-Tupel* $G = (\Sigma, H, g, \omega)$ *wobei* $H = \{\sigma_1, \dots, \sigma_n\}$ *eine endliche Menge von Produktionen ist und für alle* $1 \le i \le n$ *ist* $G_i = (\Sigma, \sigma_i, g, \omega)$ *ein* $\langle k, \ell \rangle$*-System.*

Die $\langle k, \ell \rangle$*-Systeme mit Tafeln werden auch als TIL-Systeme bezeichnet.*

Wie im kontextfreien Fall muß die Definition der erzeugten Sprache für TIL-Systeme entsprechend adaptiert werden. Wir führen dieses hier nicht explizit aus, sondern überlassen die exakte Ausformulierung dem interessierten Leser.

Definiton 4.4 *Sei G ein [T]IL-System.*

1. *G heißt* propagierend *oder* P[T]IL-System, *falls für alle Produktionen σ folgendes gilt: Wenn $(w_1, a, w_3, w_4) \in \sigma$, dann ist $w_4 \neq \lambda$.*
2. *G heißt* deterministisch *oder* D[T]IL-System, *falls für alle Produktionen σ gilt: Wenn $(w_1, a, w_3, u) \in \sigma$ und $(w_1, a, w_3, v) \in \sigma$, dann ist $u = v$.*
3. *Ist G sowohl propagierend als auch deterministisch, so ist es ein PD[T]IL-System.*

Die von G erzeugten Sprachen heißen entsprechend, d.h. P[T]IL-, D[T]IL- oder PD[T]IL-Sprache. Wenn wir etwas genauer sein wollen, und das zugrunde gelegte IL-System ein $\langle k, \ell \rangle$-System ist, sprechen wir auch von P[T]$\langle k, \ell \rangle$L-, D[T]$\langle k, \ell \rangle$L- oder PD[T]$\langle k, \ell \rangle$L-Sprachen.

Nach soviel einleitenden Definitionen ein kleines Beispiel:

Beispiel 5 Das $\langle 1, 0 \rangle$-System $G = (\{a, b, c\}, \sigma, g, c)$ mit den Produktionen:

$$\begin{array}{lcll} \langle x, a, \lambda \rangle & \to & a^2 & \text{für } x \in \{a, c, g\}, \\ \langle b, a, \lambda \rangle & \to & a, & \\ \langle x, c, \lambda \rangle & \to & ba & \text{für } x \in \{a, b, c, g\}, \\ \langle x, c, \lambda \rangle & \to & a^2 & \text{für } x \in \{a, b, c, g\} \text{ und} \\ \langle x, b, \lambda \rangle & \to & b & \text{für } x \in \{a, b, c, g\} \end{array}$$

erzeugt die Sprache

$$L(G) = \{c, ba\} \cup \{\, a^{2^n} \mid n \geq 1 \,\}.$$

Wie man leicht überprüfen kann, erzeugt das Axiom c nach einem Ableitungsschritt entweder das Wort ba oder a^2. Aus ba kann aber nur wieder ba abgeleitet werden, da nur die Produktionen $\langle g, b, \lambda \rangle \to b$ und $\langle b, a, \lambda \rangle \to a$ anwendbar sind. Es bleibt zu zeigen, daß aus a^2 alle Worte der Form a^{2^n}, für $n \geq 1$, abgeleitet werden können. Dies ist offensichtlich, da ganz allgemein für ein Wort das nur aus a's nach einem Schritt durch Anwendungen der Produktionen $\langle g, a, \lambda \rangle \to a^2$ und $\langle a, a, \lambda \rangle \to a^2$ die Anzahl der a's verdoppelt wird. Damit erhalten wir die Ableitungskette:

$$a^2 \Rightarrow a^4 \Rightarrow a^8 \Rightarrow \cdots \Rightarrow a^{2^n} \Rightarrow a^{2^{n+1}} \Rightarrow \cdots.$$

Nun zu den ersten Lemmata. Unmittelbar aus der Definition von IL-System läßt sich zum einen folgendens ableiten:

Lemma 4.1 *[P][D][T]$\langle 0, 0 \rangle$L = [P][D][T]0L.* □

Auf der anderen Seite erhalten wir trivialer Weise:

Lemma 4.2 *Seien $k_1, k_2, \ell_1, \ell_2 \in \mathbb{N}_0$. Wenn $k_1 \leq k_2$ und $\ell_1 \leq \ell_2$, dann gilt*

$$[P][D][T]\langle k_1, \ell_1\rangle L \subseteq [P][D][T]\langle k_2, \ell_2\rangle L.$$

□

Interessanter Weise kann man für [T]IL-Systeme eine spezielle Normalform bezüglich des rechten oder linken Kontexts der Produktionenmenge erzielen. Wir zeigen nämlich, daß es zu jedem IL-System ein äquivalentes $\langle 1, \ell\rangle$- bzw. $\langle k, 1\rangle$-System gibt.

Theorem 4.1 *1. Sei $k \geq 2$ und $\ell \geq 0$, dann gilt*

$$[T]\langle k, \ell\rangle L \subseteq [T]\langle k-1, \ell+1\rangle L.$$

2. Sei $k \geq 0$ und $\ell \geq 2$, dann gilt

$$[T]\langle k, \ell\rangle L \subseteq [T]\langle k+1, \ell-1\rangle L$$

Beweis. Wir beweisen nur die erste Aussage für IL-Systeme. Ein analoger Beweis kann für TIL-Systeme geführt werden. Gleiches gilt auch für die zweite Aussage.

Sei $G = (\Sigma, \sigma, g, \omega)$ ein $\langle k, \ell\rangle$-System. Wir konstruieren ein $\langle k-1, \ell+1\rangle$-System $H = (\Sigma, \pi, g, \omega)$ wie folgt.

Die Idee bei der Konstruktion ist wie folgt: Falls in G aus $a_1 \ldots a_n$ ein Wort $\alpha_1 \ldots \alpha_n$ abgeleitet werden kann, wobei jeweils $a_i \Rightarrow_G^* \alpha_i$ gilt, dann wird in H im Prinzip aus a_{i-1}, mit $2 \leq i \leq n$, das Wort α_i abgeleitet. Darüberhinaus muß a_n dann das leere Wort λ ableiten und a_1 simuliert in H das Ableiten von sowohl a_1 als auch a_2 in G. Da es bei dieser Simulation möglich sein muß, die äußersten Symbole zu finden, werden die Einschränkungen an k und ℓ benötigt.

Mit $a, b, c \in \Sigma$, $u, v \in (\Sigma \cup \{g\})^*$, $x \in \Sigma \cup \{g\}$ und $\alpha, \beta \in \Sigma^*$ sind die Produktionen von π definiert durch:

1. Mit $(g^k, a, g^\ell) \to \alpha \in \sigma$ ist $(g^{k-1}, a, g^{\ell+1}) \to \alpha \in \pi$.
2. Mit $(xub, a, g^\ell) \to \alpha \in \sigma$ ist $(ub, a, g^{\ell+1}) \to \lambda \in \pi$.
3. Mit $(ucb, a, v) \to \alpha \in \sigma$ ist $(uc, b, av) \to \alpha \in \pi$.
4. Für Regeln mit dem Linkskontext $g^{k-1}b$ unterscheiden wir zwei Fälle für ℓ:
   (a) Falls $\ell = 0$ dann ist mit $(g^{k-1}b, a, \lambda) \to \alpha \in \sigma$ und $(g^k, b, \lambda) \to \beta \in \sigma$ die Produktion $(g^{k-1}, b, a) \to \beta\alpha$ in π.
   (b) Falls $\ell > 0$ dann ist mit $(g^{k-1}b, a, ux) \to \alpha \in \sigma$ und $(g^k, b, au) \to \beta \in \sigma$ die Produktion $(g^{k-1}, b, aux) \to \beta\alpha$ in π.

Man kann leicht sehen, daß $L(G) = L(H)$ gilt. □

Aus der Existenz der Normalform erhalten wir folgendes Korollar.

Korollar 4.1 *1. Sei $k \geq 2$ und $\ell \geq 0$, dann gilt*

$$[T]\langle k, \ell\rangle L \subseteq [T]\langle 1, \ell + k - 1\rangle L.$$

2. Sei $k \geq 0$ und $\ell \geq 2$, dann gilt

$$[T]\langle k, \ell\rangle L \subseteq [T]\langle k + \ell - 1, 1\rangle L.$$

□

4.2 L-Systeme mit Hilfssymbolen und die Chomsky Hierarchie

Datum: 10. Juli

In ähnlicher Art und Weise wie im kontextfreien Fall lassen sich kontextsensitive L-Systeme mit Hilfssymbolen definieren.

Definiton 4.5 *Ein* E[T]IL-System *ist ein 5-Tupel $G = (\Sigma, H, g, \omega, \Delta)$, falls $\bar{G} = (\Sigma, H, g, \omega)$ ein [T]IL-System ist und $\Delta \subseteq \Sigma$ gilt. Die Menge Δ wird dann als* Terminalalphabet *bezeichnet.*

Die von G erzeugte Sprache wird mit $L(G)$ bezeichnet und ist definiert als

$$L(G) = L(\bar{G}) \cap \Delta^*.$$

Die bei [T]IL-Systemen eingeführten Einschränkungen D und P übertragen sich in natürlicher Weise auch auf E[T]IL-Systeme in dem man diese Eigenschaft von dem zugrundeliegenden [T]IL-System fordert. Wir sprechen dann von ED[T]IL-, E[P]TIL- und EPD[T]IL-Systemen und Sprachen.

Im Gegensatz zu kontextfreien L-Systeme fallen kontextsentive L-Systeme mit Hilfssymbolen im wesentlichen mit den drei bekannten Sprachklassen DSpace(n), CS und RE zusammen. Da die Ergebnisse im wesentlichen auf einfachen Turingmaschinen Simulationen beruhen, werden wir diese hier nicht ausführen, und verweisen stattdessen auf die Arbeiten von van Dalen [12], Vitanyi [15, 16] und Wood [18].

Theorem 4.2 *1. $DSpace(n) = EPDIL$.*

2. $CS = EPIL = EPTIL = EPDTIL$.

3. $RE = EIL = ETIL = EDIL = EDTIL$. □

Wir schließen die ergänzende Vorlesung zu „Formalen Sprachen“ mit der Bemerkung, daß angesichts der Darstellung der Sprachfamilien DSpace(n) und CS durch kontextsensitive L-Systeme die sogenannte LBA-Frage von Kuroda [4] alternative formulieren läßt. Kurodas Frage war, ob sich jede Sprache die durch eine nichtdeterminitische linear beschränkte Turingmachine akzeptiert wird auch durch eine deterministische linear beschränkten Turingmaschine akzeptiert wird. Das Kürzel LBA kommt von den englischen Worten *linear bounded automaton*. Diese Frage ist im Bereich der Komplexitätstheorie eine der vielen ungelösten

Fragen zur Beziehung von deterministischen und nichdeterministischen Komplexitätsklassen.

Im Sinne von kontextsensitiven L-Systemen stellt sich also die Frage „EPDIL = EPIL?“, oder mit anderen Worten, „Erhöhen λ-Produktionen die Erzeugsmächtigkeit von EPDIL-Systemen?“ Obwohl diese Frage recht harmlos aussieht scheint sie doch recht kompliziert zu sein, da sich mit ihr ein lange offenes Problem lösen würde. OK ←

5 Danksagung und abschließende Worte

Mein besonderer Dank gilt Jörg Hoss, der einen Großteil dieses Skripts getippt hat und mich im Sommersemester 1997 bei der Übungsleitung zur Vorlesung „Formale Sprachen“ tatkräftig unterstützt hat. Auf diesem Wege auch Dank an Martin Böhler für die Unterstützung bei dieser Arbeit und an Michael Hauth sowie Stefan Gumhold für das Korrekturlesen des Skripts.

Bei eventuellen Fragen oder Anmerkungen (Kommentare, Darstellung, Fehler, usw.) schicken Sie mir bitte eine email an die Adresse

`holzer@informatik.uni-tuebingen.de`

Im übrigen kann für die Richtigkeit des Inhalts dieses Skripts keine Haftung übernommen werden.

Literatur

[1] C. Damm, M. Holzer, and K.-J. Lange. The parallel complexity of iterated morphisms and the arithmetic of small numbers. In *Proceedings of the 17th Conference on Mathematical Foundations of Computer Science*, number 629 in LNCS, pages 227–235. Springer, August 1992.

[2] C. Damm, M. Holzer, K.-J. Lange, and P. Rossmanith. Deterministic 0L languages are of very low complexity : D0L is in AC^0. In *Developments in Language Theory*, pages 305–313. World Scientific, Singapore, 1994.

[3] G. T. Herman and G. Rozenberg. *Developmental Systems and Languages.* North-Holland/American Elsevier, 1975.

[4] S.-Y. Kuroda. Classes of languages and linear bounded automata. *Information and Control*, 7:207–223, 1964.

[5] K.-J. Lange. Equivalence of adult languages and extensions for DT0L systems. *Information and Control*, 47:107–112, 1980.

[6] A. Lindenmayer. Mathematical models for cellular interactions in develo pment, II. Simple and branching filaments with two-sided inputs. *J. theoret. Biol.*, 18:300–315, 1968.

[7] A. Lindenmayer. Mathematical models for cellular interactions in development, I. Filaments with one-sided inputs. *J. theoret. Biol.*, 18:280–299, 1968.

[8] G. Rozenberg. On a family of acceptors for some classes of developmental languages. *International Journal of Computer Mathematics*, 4:199–228, 1974.

[9] G. Rozenberg and A. Salomaa. *The Mathematical Theory of L Systems*, volume 90 of *Pure and Applied Mathematics*. Academic Press, 1980.

[10] G. Rozenberg and D. Wood. A note on a family of acceptors of developmental languages. *International Journal of Computer Mathematics*, A-5:261–266, 1976.

[11] I. H. Sudborough. On the tape complexity of deterministic context-free languages. *Journal of the ACM*, 25:405–414, 1978.

[12] D. van Dalen. A note on some systems of Lindenmayer. *Mathematical Systems Theory*, 5:128–150, 1971.

[13] J. van Leeuwen. *Notes on pre-set pushdown automata*, volume 15 of *LNCS*, pages 177–188. Springer, 1974.

[14] J. van Leeuwen. Variations of a new machine model. In *Proceedings of the 17th Annual Symposium on Foundations of Computer Science*, pages 228–235, Houston, Texas, 1976.

[15] P. M. B. Vitányi. Deterministic Lindenmayer languages, nonterminals and homomorphisms. *Theoretical Computer Science*, 2:49–71, 1976.

[16] P. M. B. Vitányi. Context senstive table Lindenmayer languages and a relation to the LBA problem. *Information and Control*, 33:217–226, 1977.

[17] P. M. B. Vitányi and A. Walker. Stable string languages of Lindenmayer systems. *Information and Control*, 37(2):134–149, May 1978.

[18] D. Wood. Iterated a-NGSM maps and L-systems. *Information and Control*, 32:1–26, 1975.