Automatentheorie und Formale Sprachen

Vorlesungsskript

Dietmar Wätjen

Institut für Theoretische Informatik
Technische Universität Braunschweig

1999

Inhaltsverzeichnis

Einleitung

Die Vorlesung „Automatentheorie und Formale Sprachen“ gehört in das Gebiet der Theoretischen Informatik. Die Theoretische Informatik hatte ihre Anfänge in verschiedenen Gebieten:

- Biologen studierten Modelle für Neuronennetze,
- Elektroingenieure entwickelten die Netzwerktheorie als ein Mittel zur Hardwarebeschreibung,
- Mathematiker arbeiteten über Grundlagen der Logik und
- Linguisten untersuchten Grammatiken für natürliche Sprachen.

Aus diesen Studien ergaben sich Modelle, die für die Theoretische Informatik von großer Bedeutung sind und die auch in dieser Vorlesung betrachtet werden.

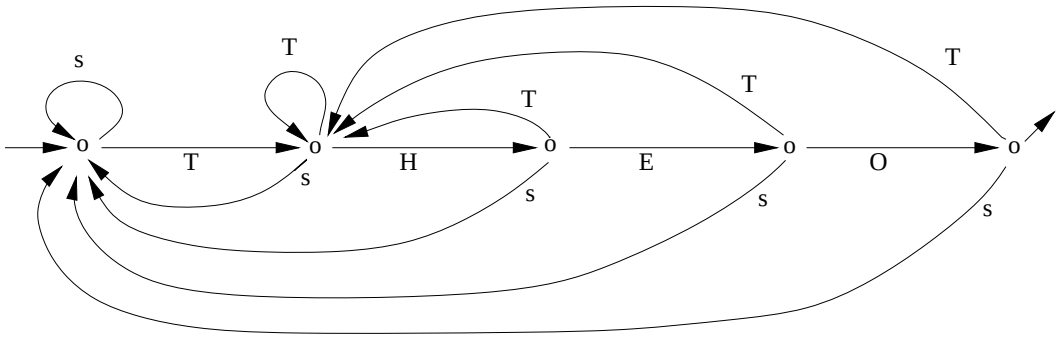
Vielfach wird gefragt, was eigentlich der Nutzen der Theoretischen Informatik ist. Es wird bezweifelt, ob es sich dabei überhaupt um „richtige“ Informatik handelt, und es wird kritisiert, daß die Vorlesungen der Theoretischen Informatik zu mathematisch sind, also viel zu abstrakt. Man kann wohl nicht erwarten, daß die Mehrheit der Informatikstudenten die Theoretische Informatik so schätzen wird wie z.B. das Programmieren. Aber Informatik ist nicht nur Programmieren und braucht eine mathematische Fundierung. Der Eindruck der Unverständlichkeit und des eigentlich Unnötigen, den viele von der Theoretischen Informatik haben, wird sich hoffentlich auch für Sie als irrig herausstellen, wenn Sie im Laufe Ihres Studiums die Beziehungen zwischen Theorie und Praxis erkennen. Gewisse Gegenstände, Ideen, Beweistechniken, die in der Theoretischen Informatik benutzt werden (z.B. Keller), hat man zum Teil schon vorher in Gebieten der Praktischen Informatik ebenso oder wenigstens ähnlich gesehen bzw. wird sie dort noch kennenlernen.

Es sollen hier schon einige der in der Vorlesung behandelten Gebiete mit ihren Berührungspunkten zur Praxis aufgezählt werden. So werden in dieser Vorlesung endliche Automaten und reguläre Ausdrücke behandelt. Sie wurden ursprünglich entwickelt, um Neuronennetze und Netzwerke modellieren zu können. Später hat man festgestellt, daß sie z.B. auch als nützliche Hilfsmittel zur lexikalischen Analyse in einem Compiler dienen können. Dabei werden *characters* in unzerlegbare Einheiten gruppiert, also in Variablennamen und Schlüsselwörter. Weitere Anwendungen für reguläre Ausdrücke und endliche Automaten wurden in Texteditoren, im Pattern-Matching (Übereinstimmung von Mustern) und in verschieden Textverarbeitungs- und File-Suchprogrammen gefunden. Unter dem Betriebssystem UNIX gibt es z.B. die Möglichkeit, Files nach Wörtern aus einem regulären Ausdruck zu durchsuchen, also festzustellen, ob im Text des Files entsprechende Wörter vorkommen. Als Beispiel betrachten wir den einfachen

regulären Ausdruck

$$\{THEO\}.$$

Zunächst wird in einem Vorlauf ein zugehöriger endlicher Automat konstruiert:



s = sonstige Buchstaben

Der Automat erhält als Eingabe einen Text. Wird $THEO$ als Teil dieses Textes gefunden, so geht der Automat in den durch einen ausgehenden Pfeil ohne Ziel gekennzeichneten Endzustand über. Das kann dazu führen, daß ein Editor hinter dem entsprechenden Wort stehenbleibt oder auch die entsprechende Zeilennummer des Vorkommens ausgedruckt wird. Die Konstruktion kann auf allgemeine reguläre Ausdrücke erweitert werden. Abgesehen vom Vorlauf ist die Zeitkomplexität linear abhängig von der Länge des Textes. Bei einem unmittelbaren Vergleich des Textes mit dem Schlüsselwort ist dagegen die Zeitkomplexität durch das Produkt der Länge des Textes mit der Länge des Schlüsselwortes gegeben und somit größer.

Weiter werden wir in der Vorlesung auf den Begriff der kontextfreien Sprachen sowie der zugehörigen Kellerautomaten zu sprechen kommen. Sie haben zur Beschreibung von Programmiersprachen und zum Entwurf von Parsern, einem anderen Teil eines Compilers, beigetragen. So ist z.B. Algol 60 durch eine kontextfreie Grammatik erklärt und wird dann durch semantische Zusatzregeln in natürlicher Sprache weiter eingeschränkt. Ähnliches gilt für Modula-2. Der Zusammenhang eines Kellerautomaten mit der Datenstruktur eines Kellers, die bei einem Compiler benötigt wird, ist unmittelbar erkennbar.

Das Problem der Berechenbarkeit hängt mit Turingmaschinen zusammen. Sie wissen vielleicht aus den Anfängervorlesungen, daß es Funktionen gibt, die nicht berechenbar sind. Ein typischer Vertreter eines nicht berechenbaren Problems ist das Halteproblem. Bezogen auf Programme bedeutet dieses: Es gibt kein Programm, das jemals geschrieben werden könnte, das für jedes beliebig vorgegebene Argument eines beliebigen anderen Programms entscheidet, ob dieses Programm mit diesem Argument irgendwann einmal hält oder nicht.

Ein wichtiger Begriff in der Informatik ist der der Simulation. Im Rahmen der Chomsky-Hierarchie werden wir die Simulation von Grammatiken durch Maschinen bzw. Automaten und umgekehrt betrachten. Simulation gibt es z.B. in der realen Welt

als Emulation eines Rechners durch einen anderen. Ein LISP-Programm, das Zeile für Zeile interpretiert wird, wird durch die unterliegende Maschinensprache simuliert.

Überhaupt ist Theorie im weitesten Sinn wichtig. So hat *D. Knuth* bei der Entwicklung des METAFONT-Programms im Rahmen des TEX-Systems, des bekannten Textsystems für wissenschaftliche Texte (mit dessen Hilfe auch dieses Skript geschrieben ist), digitale Algorithmen für trigonometrische Funktionen entwickelt. Damit können dann schließlich gewisse Bézier-Spline-Kurven gezeichnet werden, aus denen sich die Buchstaben zusammensetzen, die METAFONT produziert. Sie sehen also: auch bei praktischen Problemen kommt man ohne theoretische Kenntnisse nicht weit.

In der Vorlesung werden wir auf die oben angesprochenen Anwendungen nicht eingehen. Das gehört in entsprechende andere Vorlesungen und wird dann hoffentlich zu einigen Aha-Effekten führen. Wir werden hier die mathematische Theorie der Automaten und formalen Sprachen darstellen.

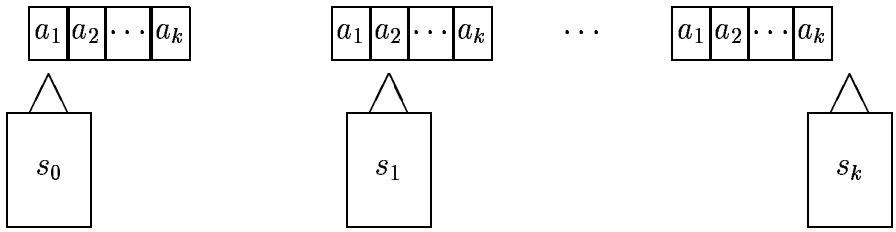
Bei der Ausarbeitung der Vorlesung wurde vor allem das Buch von *A. Salomaa* (siehe [23] bzw. [24]) zu Rate gezogen, das einen großen Einfluß auf weite Teile der Vorlesung hat. Daneben sind im Literaturverzeichnis noch weitere Bücher genannt, die zu einer vertiefenden Beschäftigung mit der Theorie der Automaten und formalen Sprachen geeignet sind.

Kapitel 1

Endliche erkennende Automaten

1.1 Definitionen

Endliche erkennende Automaten können Eingabewörter über einem festen Zeichenvorrat voneinander unterscheiden. Die Arbeitsweise wird durch die folgende Skizze verdeutlicht:



Dabei sei s_0 der Anfangszustand des Automaten, $s_1, \ldots, s_k$ seien weitere Zustände, und $a_1 \ldots a_k$ sei das Eingabewort, das von links nach rechts abgearbeitet wird. Mit jedem Lesen eines Symbols a_i, $i = 1, \ldots, k$, findet eine Zustandsänderung von s_{i-1} in s_i statt. Es sei S_1 eine ausgezeichnete Menge von Endzuständen. Nach Abarbeitung von $a_1 \ldots a_k$ befinde sich der Automat im Zustand s_k. Gilt $s_k \in S_1$, so sagt man, daß der Automat das Wort $a_1 \ldots a_k$ akzeptiert, anderenfalls ist das Wort nicht akzeptiert worden. Äquivalent können wir uns die Arbeitsweise des Automaten auch so veranschaulichen, daß die Symbole a_1 bis a_k in dieser Reihenfolge zu diskreten Zeitabständen eingegeben werden und jeweils eine entsprechende Zustandsänderung veranlassen. Bevor wir endliche erkennende Automaten definieren, führen wir noch formal die Begriffe Wort, freies Monoid, Alphabet und formale Sprache ein.

Definition 1.1 (a) Es sei A eine endliche nichtleere Menge. Dann heißt $A^* = \{x_1 \ldots x_n \mid x_i \in A \ , \ 1 \leq i \leq n \ , \ n \in \mathbb{N}_0\}$ die *Menge der Wörter über* A. $w = x_1 \ldots x_n$ heißt *Wort über* A, es hat die *Länge* $|w| = n$. Für $n = 0$ erhalten wir das *leere Wort* ε mit $|\varepsilon| = 0$.

(b) Es seien $w_1 = x_1 \dots x_n$ und $w_2 = y_1 \dots y_m$ zwei Wörter über A. Dann wird $w_1 w_2 = x_1 \dots x_n y_1 \dots y_m$ die *Konkatenation* von w_1 und w_2 genannt.
(c) Es werde $A^+ = A^* - \{\varepsilon\}$ gesetzt. □

Aus der Algebra ist bekannt, daß eine Menge $H \neq \emptyset$ mit einer Verknüpfung $\circ : H \times H \to H$ *Halbgruppe* heißt, wenn für alle $h_1, h_2, h_3 \in H$ das Assoziativgesetz

$$(h_1 \circ h_2) \circ h_3 = h_1 \circ (h_2 \circ h_3)$$

gilt. Eine Halbgruppe ist ein *Monoid*, wenn zusätzlich ein Einselement $e \in H$ existiert mit

$$e \circ h = h$$

für alle $h \in H$. Wir stellen fest, daß A^* ein Monoid ist, wenn wir als Verknüpfung die Konkatenation und als Einselement das leere Wort ε wählen. Wir nennen A^* auch *freies Monoid* über A. A^+ ist eine Halbgruppe, die *freie Halbgruppe* über A.

Definition 1.2 Eine endliche Menge V heißt *Alphabet*. Eine Teilmenge $L \subset V^*$ heißt *(formale) Sprache über* V. Es seien L_1 und L_2 beliebige Sprachen. Dann ist ihre *Konkatenation* definiert durch

$$L_1 L_2 = \{w \mid w = w_1 w_2,\ w_1 \in L_1, w_2 \in L_2\}. \quad \square$$

Beispiel 1.1 Es sei $V = \{$A, Aachen, Aal, ..., Zytode, z.Z., ., ..., :$\}$ (Einträge im Duden). Dann ist die deutsche Sprache eine Sprache $L \subset V^*$. Eine endliche Sprache über V ist z.B. $L = \{$ich habe Hunger, ich habe Durst, gute Nacht$\}$, die nur aus drei „Wörtern“ über V besteht. □

Definition 1.3 $E = (V, F)$ heißt (*deterministischer*) *endlicher erkennender Automat*, wenn die folgenden Eigenschaften erfüllt sind:
(a) Es gilt $V = S \cup V_T$ mit *Zustandsalphabet* bzw. *Zustandsmenge* S und *Eingabealphabet* V_T, wobei $S \cap V_T = \emptyset$ ist.
(b) Ein *Anfangszustand* $s_0 \in S$ und eine *Menge von Endzuständen* S_1 werden ausgezeichnet.
(c) F ist eine endliche Menge von Paaren $(s_i a_k, s_j)$ mit $s_i, s_j \in S$, $a_k \in V_T$, die *Ersetzungs-* oder *Überführungsregeln* oder *Produktionen* heißen.
(d) Für jedes $(s_i, a_k) \in S \times V_T$ existiert genau ein $s_j \in S$ mit $(s_i a_k, s_j) \in F$. □

Statt $(s_i a_k, s_j) \in F$ schreiben wir auch $s_i a_k \to_E s_j$ oder, falls keine Mißverständnisse auftreten können, $s_i a_k \to s_j$. Man beachte, daß das Zustands-, das Eingabealphabet, der Anfangsszustand und die Menge der Endzustände nicht explizit in der Schreibweise $E = (V, F)$ berücksichtigt werden. Schreiben wir nur E oder $E = (V, F)$, so unterstellen wir in der Regel, daß die einzelnen Größen wie in Definition 1.3 notiert sind. Oft wird ein solcher endlicher erkennender Automat auch explizit als 5-Tupel

$(S, V_T, \delta, s_0, S_1)$ geschrieben (siehe z.B. [30], Definition 5.11), wobei $\delta : S \times V_T \to S$ die Überführungsfunktion ist. Es gilt dann

$$\delta(s_i, a_k) = s_j \iff (s_i a_k, s_j) \in F.$$

Wir haben hier endliche erkennende Automaten in der Form von Definition 1.3 angegeben, weil sie sich so als spezielle kombinatorische Systeme auffassen lassen, die wir in Definition 1.4 allgemein definieren werden. Auch andere Automaten- und Maschinentypen werden später als solche Systeme eingeführt, und Grammatiken sind ebenfalls spezielle kombinatorische Systeme.

Definition 1.4 $KS = (V, F)$ heißt *kombinatorisches System*, wenn
(a) V ein Alphabet und
(b) F eine endliche Teilmenge von $\{(P, Q) \mid P, Q \in V^*\}$ ist.
$(P, Q) \in F$ heißt *Ersetzungsregel* oder *Produktion*. Man schreibt auch $P \to Q$. □

Definition 1.5 Es sei $KS = (V, F)$ ein kombinatorisches System.
(a) $P \in V^*$ *erzeugt direkt* $Q \in V^*$ (kurz $P \Longrightarrow_{KS} Q$ oder $P \Longrightarrow Q$), wenn $P', P_1, P'', Q_1 \in V^*$ existieren mit $P = P'P_1P''$, $Q = P'Q_1P''$ und $(P_1, Q_1) \in F$.
(b) $P \in V^*$ *erzeugt* $Q \in V^*$ (kurz $P \Longrightarrow_{KS}^* Q$ oder $P \Longrightarrow^* Q$), wenn $P_0, P_1, \ldots, P_k \in V^*$ existieren mit $k \in \mathbb{N}_0$, $P_0 = P$, $P_k = Q$ und $P_i \Longrightarrow P_{i+1}$ für $0 \leq i \leq k-1$.
Die Folge $P_0, P_1, \ldots, P_k$ heißt *Ableitung von Q aus P in KS*. Man schreibt auch $P_0 \Longrightarrow P_1 \Longrightarrow \ldots \Longrightarrow P_k$, wobei k als *Länge der Ableitung* bezeichnet wird. □

Offenbar ist $\Longrightarrow$ eine binäre Relation über V^*. Dann ist $\Longrightarrow^*$ die reflexive, transitive Hülle von $\Longrightarrow$. Allgemein gilt die folgende

Definition 1.6 Es sei ρ eine binäre Relation über einer Menge M. Die *reflexive, transitive Hülle* ρ^* von ρ ist wie folgt gegeben:
(a) $\bigwedge_{P \in M} P\rho^*P \wedge \bigwedge_{P,Q \in M} (P\rho Q \Longrightarrow P\rho^*Q)$.
(b) $\bigwedge_{P_1,P_2,P_3 \in M} (P_1\rho^*P_2 \wedge P_2\rho^*P_3 \Longrightarrow P_1\rho^*P_3)$.
(c) $P\rho^*Q \iff$ Ausdruck nach (a) und (b) aufgebaut. □

Wir weisen darauf hin, daß wir in dieser Definition das Symbol $\Longrightarrow$ für die Implikation verwendet haben. $\Longrightarrow$ wird also sowohl in dieser logischen Bedeutung als auch für einen Ableitungsschritt benutzt. Dies wird jedoch zu keinen Verwechslungen führen, aus dem Kontext wird immer die jeweilige Bedeutung erkennbar sein.

Für endliche erkennende Automaten erhalten wir die folgende Definition.

Definition 1.7 Es sei E ein endlicher erkennender Automat.
(a) $L(E) = \{w \in V_T^* \mid s_0 w \Longrightarrow^* s_1,\ s_1 \in S_1\}$ heißt die *von E angenommene* oder *erkannte (akzeptierte) Sprache*.

(b) $L \subset V_T^*$ heißt *von einem endlichen erkennenden Automaten erkennbar (akzeptierbar)*, wenn ein endlicher erkennender Automat E mit $L(E) = L$ existiert. □

Man beachte, daß für endliche erkennende Automaten die Relation $\Longrightarrow^*$ nur sehr speziell genutzt wird. Solange für $P \Longrightarrow^* Q$ nicht $P \in V_T^*$ gilt, ist $P \in SV_T^*$.

Definition 1.8 Ein kombinatorisches System $N = (V, F)$ heißt *nichtdeterministischer endlicher erkennender Automat*, wenn die folgenden Eigenschaften erfüllt sind:

(a) Es gilt $V = S \cup V_T$ mit *Zustandsalphabet* bzw. *Zustandsmenge* S und *Eingabealphabet* V_T, wobei $S \cap V_T = \emptyset$ ist.

(b) Eine *Menge von Anfangszuständen* $S_0 \subset S$ und eine *Menge von Endzuständen* $S_1 \subset S$ werden ausgezeichnet.

(c) F ist eine endliche Menge von Paaren $(s_i a_k, s_j)$ mit $s_i, s_j \in S$, $a_k \in V_T$, die *Ersetzungs-* oder *Überführungsregeln* oder *Produktionen* heißen. □

Definition 1.9 Es sei $N = (V, F)$ ein nichtdeterministischer endlicher erkennender Automat.

(a) $L(N) = \{w \in V_T^* \mid s_0 w \Longrightarrow^* s_1,\ (s_0, s_1) \in S_0 \times S_1\}$ heißt die *von N angenommene* oder *erkannte (akzeptierte) Sprache.*

(b) $L \subset V_T^*$ heißt *von einem nichtdeterministischen endlichen erkennenden Automaten erkennbar (akzeptierbar)*, wenn ein nichtdeterministischer endlicher erkennender Automat N mit $L(N) = L$ existiert. □

Wir sehen, daß (a) und (c) der Definitionen 1.3 und 1.8 übereinstimmen. Die Bedingung (d) fehlt in Definition 1.8, woraus sich der Nichtdeterminismus der entsprechenden Automaten ergibt. Man beachte weiter, daß nichtdeterministische Automaten eine Menge von Anfangszuständen und nicht nur einen einzigen Anfangszustand besitzen. Dies hat einige beweistechnische Vorteile, obwohl es an den prinzipiellen Fähigkeiten der nichtdeterministischen endlichen erkennenden Automaten nichts ändert. Auf dem ersten Blick möchte man nichtdeterministische Automaten für „mächtiger" als deterministische Automaten halten. Die „Annahmekapazitäten" beider Typen von Automaten sind jedoch gleich.

Satz 1.1 Es sei $N = (V, F)$ ein nichtdeterministischer endlicher erkennender Automat. Dann existiert ein deterministischer endlicher erkennender Automat E mit $L(N) = L(E)$.

Beweis: Wir nehmen an, daß N durch die Größen aus Definition 1.8 gegeben ist. Die Grundidee ist, daß die Elemente der Potenzmenge $\mathfrak{P}(S)$ der Zustandsmenge S des Automaten N, also die Teilmengen von S, als Zustände von E aufgefaßt werden. Deshalb heißt der Automat E auch *Potenzautomat* von N.

Wir konstruieren $E = (V', F')$ wie folgt. Wir setzen

$$V' = \mathfrak{P}(S) \cup V_T,$$

wobei wir $\bar{S} = \mathfrak{P}(S)$ abkürzen. Es wird

$$S_0 \in \bar{S} \text{ als Anfangszustand von } E$$

und

$$\bar{S}_1 = \{S' \mid S' \in \bar{S},\ S' \cap S_1 \neq \emptyset\} \text{ als Menge der Endzustände}$$

gewählt. Die Produktionenmenge wird schließlich dadurch gegeben, daß wir allen $S'_k \in \bar{S}$, $x \in V_T$ genau die Produktion $(S'_k x, S'_l) \in F'$ zuordnen mit

$$S'_l = \{s \mid (\bar{s}x, s) \in F,\ \bar{s} \in S'_k\}.$$

Den Beweis wollen wir hier nicht weiter ausführen. Er befindet sich z.B., allerdings in anderer Notation, in [30], S. 122/123. □

Ein weniger direkter Beweis dieses Satzes wird auch durch den Beweis von Satz 3.3 geliefert. Man beachte, daß die Umkehrung dieses Satzes aufgrund der Definitionen trivial ist. Deterministische und nichtdeterministische endliche erkennende Automaten sind also im Sinne der folgenden Definition äquivalent.

Definition 1.10 Es seien E und E' deterministische oder nichtdeterministische endliche erkennende Automaten. Sie heißen *äquivalent*, wenn $L(E) = L(E')$ gilt. □

Dies ist eine „grobe" Ergebnisäquivalenz, da es nur darauf ankommt, ob man bei einem eingegebenen Wort vom Anfangszustand in irgendeinen Endzustand gelangt oder nicht.

1.2 Abschlußoperationen bei erkannten Sprachen

Satz 1.2 Es seien $E = (V, F)$ und $E' = (V', F')$ endliche erkennende Automaten. Dann existieren endliche erkennende Automaten E_V, E_P, E_I, E_K und E_D mit den folgenden Eigenschaften:

(a) $L(E_V) = L(E) \cup L(E')$,
(b) $L(E_P) = L(E)L(E')$,
(c) $L(E_I) = (L(E))^*$,
(d) $L(E_K) = V_T''^* - L(E)$ für jedes Alphabet V_T'',
(e) $L(E_D) = L(E) \cap L(E')$.

Beweis: E sei durch die Daten S, V_T, s_0 und S_1 gegeben, E' entsprechend in „gestrichener" Version. Ohne Beschränkung der Allgemeinheit gelte $S \cap S' = \emptyset$. Es wird jeweils ein nichtdeterministischer endlicher erkennender Automat $\bar{N} = (\bar{V}, \bar{F})$ konstruiert, der die entsprechende Sprache erkennt. Aufgrund von Satz 1.1 existiert dann auch ein deterministischer endlicher erkennender Automat mit denselben Eigenschaften.

(a) Wir setzen $\bar{V} = V \cup V'$, $\bar{S}_0 = \{s_0, s'_0\}$, $\bar{S}_1 = S_1 \cup S'_1$, $\bar{F} = F \cup F'$. Der Nichtdeterminismus liegt nur in der Wahl des Anfangszustandes s_0 oder s'_0. Offenbar gilt $L(\bar{N}) = L(E) \cup L(E')$.

(b) Wir setzen $\bar{V} = V \cup V'$ und definieren

$$\bar{F} = \{(s_i x, s_j) \mid (s_i x, s_j) \in F \cup F'\} \cup \{(s_i x, s_0') \mid (s_i x, s_j) \in F, s_j \in S_1\},$$

$$\bar{S}_0 = \begin{cases} \{s_0, s_0'\}, & \text{falls } \varepsilon \in L(E) \\ \{s_0\} & \text{sonst} \end{cases}$$

und $\bar{S}_1 = S_1'$. Dabei ist ε das leere Wort. Es gilt $L(\bar{N}) = L(E)L(E')$.

(c) Wir setzen $\bar{S} = S \cup \{a\}$ mit $a \notin V_T \cup S$, $\bar{V}_T = V_T$, $\bar{S}_0 = \{s_0, a\}$ und $\bar{S}_1 = S_1 \cup \{a\}$. Weiter werde

$$\bar{F} = \{(s_i x, s_j) \mid (s_i x, s_j) \in F\} \cup \{(s_i x, s_0) \mid (s_i x, s_j) \in F, s_j \in S_1\}$$

gesetzt. Das Hilfssymbol a ist nötig, um das leere Wort ε zu erzeugen. Dieses Symbol kommt in $\bar{F}$ nicht vor, was zulässig ist, da ein nichtdeterministischer endlicher erkennender Automat nicht vollständig sein muß. Wir erhalten $L(\bar{N}) = (L(E))^*$.

(d) Wir setzen $\bar{V}_T = V_T''$, $\bar{S} = S \cup \{t\}$ (mit neuem Symbol t), $\bar{S}_0 = \{s_0\}$, $\bar{S}_1 = \bar{S} - S_1$ und

$$\begin{aligned} \bar{F} \;=\; & \{(s_i x, s_j) \mid (s_i x, s_j) \in F, x \in V_T'' \cap V_T\} \cup \{(s_i x, t) \mid s_i \in S, x \in V_T'' - V_T\} \\ & \cup \{(tx, t) \mid x \in V_T''\}. \end{aligned}$$

Man beachte, daß im Fall $V_T'' \subset V_T$ die zweite der Teilmengen von $\bar{F}$ leer ist. Wörter mit Symbolen aus $V_T'' - V_T$ werden wegen des Endzustands t immer akzeptiert. Somit geht der (deterministische) Automat $\bar{N}$ genau dann in einen Endzustand über, wenn E in einen Nichtendzustand übergeht oder wegen des Vorkommens eines Symbols aus $V_T'' - V_T$ gar nicht arbeiten kann.

(e) Man wähle $V_T'' = V_T \cup V_T'$ und bezeichne das Komplement einer Menge M bezüglich $V_T''^*$ durch $\complement M$. Dann gilt

$$L(E) \cap L(E') = \complement\complement L(E) \cap \complement\complement L(E') = \complement(\complement L(E) \cup \complement L(E')).$$

Somit folgt (e) aus (a) und (d). □

Wir sagen auch, daß die Familie der Sprachen, die durch endliche erkennende Automaten erkannt werden, *abgeschlossen* ist unter Vereinigungs-, Durchschnitts- und Komplementbildung sowie unter Konkatenation und Iteration.

1.3 Reduktion von deterministischen Automaten

Definition 1.11 Es sei $E = (V, F)$ ein endlicher erkennender Automat. Ein Zustand $s \in S$ heißt *erreichbar*, wenn ein $w \in V_T^*$ existiert mit $s_0 w \Longrightarrow^* s$. s heißt *unerreichbar*, wenn er nicht erreichbar ist. E heißt *vereinfacht*, wenn alle Zustände $s \in S$ erreichbar sind. □

Satz 1.3 Es sei $E = (V, F)$ ein endlicher erkennender Automat. Dann existiert ein vereinfachter endlicher erkennender Automat E' mit $L(E) = L(E')$, der in endlich vielen Schritten konstruiert werden kann.

Beweis: Es sei n die Anzahl der Zustände aus S. Wir zeigen zunächst, daß es für jeden erreichbaren Zustand $s \in S$ ein $w \in V_T^*$, $|w| < n$, gibt mit $s_0 w \Longrightarrow^* s$. Es sei nämlich

$$s_0 v \Longrightarrow^* s \text{ mit } v = a_1 \dots a_m,\ m \geq n.$$

Dann wird bei der Abarbeitung von v die Zustandsfolge

$$s_0,\ s_1 \text{ mit } s_0 a_1 \to s_1,\ s_2 \text{ mit } s_1 a_2 \to s_2,\ \dots,\ s_m = s \text{ mit } s_{m-1} a_m \to s_m$$

durchlaufen. Wegen $m \geq n$ tritt wenigstens ein Zustand in dieser Folge mindestens zweimal auf, etwa $s_i = s_j$, $0 \leq i < j \leq m$. Folglich wird bei Abarbeitung des Wortes $v' = a_1 \dots a_i a_{j+1} \dots a_m$ die Zustandsfolge

$$s_0, \dots, s_i = s_j, \dots, s_m = s$$

durchlaufen. Dabei gilt $s_0 v' \Longrightarrow^* s$ und $|v'| < |v|$. Dieser Prozeß wird (endlich oft) wiederholt, bis sich ein Wort $w \in V_T^*$ der gewünschten Gestalt ergibt.

Die Menge $\{s \mid s_0 w \Longrightarrow^* s, w \in V_T^*, |w| < n\}$ besteht also aus genau allen erreichbaren Zuständen und ist in endlich vielen Schritten konstruierbar. Da die nicht erreichbaren Zustände überflüssig sind, liefert ihre Entfernung aus S und S_1 sowie eine entsprechende Änderung der Produktionenmenge F einen vereinfachten endlichen erkennenden Automaten mit den oben angegebenen Eigenschaften. □

Definition 1.12 Es sei $E = (V, F)$ ein endlicher erkennender Automat.

(a) Es sei $s \in S$. Dann ist E_s der endliche erkennende Automat, der wie E definiert ist, jedoch s als Anfangszustand besitzt.

(b) Es seien $s', s'' \in S$. Die Zustände s', s'' heißen *äquivalent* (in Zeichen $s' \sim_E s''$ oder $s' \sim s''$), wenn $L(E_{s'}) = L(E_{s''})$ gilt. Die Zustände s', s'' heißen *inäquivalent*, wenn s', s'' nicht äquivalent sind. □

Aus der Definition folgt unmittelbar

Satz 1.4 Es sei E ein endlicher erkennender Automat und $s', s'' \in S$. Dann gilt

$$s' \sim s'' \iff \bigwedge_{w \in V_T^*} \bigvee_{\bar{s}', \bar{s}'' \in S} ((s'w \Longrightarrow^* \bar{s}' \in S_1) \iff (s''w \Longrightarrow^* \bar{s}'' \in S_1)). \quad \square$$

Wir erkennen sofort, daß die Relation $\sim$ eine Äquivalenzrelation auf der Zustandsmenge S ist. Daher ist die nächste Definition sinnvoll.

Definition 1.13 Es sei E ein endlicher erkennender Automat und $s \in S$. Mit $[s] \subset S$ bezeichnen wir die durch $\sim$ erzeugte *Äquivalenzklasse von s.* □

Definition 1.14 Es sei E ein endlicher erkennender Automat. E heißt *reduziert*, wenn E vereinfacht ist und die Zustände von E paarweise inäquivalent sind. □

Satz 1.5 Es sei E ein vereinfachter endlicher erkennender Automat. Dann existiert ein zu E äquivalenter reduzierter endlicher erkennender Automat E', wobei für die Zustandsmengen $\text{card}(S) \geq \text{card}(S')$ gilt.

Beweis: Wir konstruieren den Automaten E' durch

$$S' = \{[s] \mid s \in S\},\ s'_0 = [s_0],\ S'_1 = \{[s] \mid s \in S_1\}$$

und

$$F' = \{([s]a, [s']) \mid (sa, s') \in F, s, s' \in S, a \in V_T\}.$$

Wir zeigen zunächst, daß die Menge F' wohldefiniert ist. Es sei $r \in [s]$, d.h., es gelte $r \sim s$. Zu zeigen ist, daß für jedes $a \in V_T$ aus $(ra, r') \in F$ und $(sa, s') \in F$ die Relation $r' \sim s'$ folgt, also $r' \in [s']$ gilt. Nach Satz 1.4 erhalten wir

$$r \sim s \iff \bigwedge_{w \in V_T^*} \bigvee_{\bar{r}, \bar{s} \in S} ((rw \Longrightarrow^* \bar{r} \in S_1) \iff (sw \Longrightarrow^* \bar{s} \in S_1)).$$

Insbesondere folgt dann mit dem obigen $a \in V_T$

$$\bigwedge_{w \in V_T^*} \bigvee_{\bar{r}, \bar{s} \in S} ((raw \Longrightarrow r'w \Longrightarrow^* \bar{r} \in S_1) \iff (saw \Longrightarrow s'w \Longrightarrow^* \bar{s} \in S_1)).$$

Lassen wir in beiden Ableitungen den ersten Ableitungsschritt weg, so erhalten wir nach Satz 1.4 die Relation $r' \sim s'$.

Für $s \in S$, $w \in V_T^*$ mit $s_0 w \Longrightarrow_E^* s$ folgt offenbar $[s_0]w \Longrightarrow_{E'}^* [s]$, d.h., mit E ist auch E' vereinfacht. Nach Konstruktion von E' gilt weiter $\text{card}(S) \geq \text{card}(S')$.

Zum Beweis der Äquivalenz von E und E' zeigen wir allgemeiner $L(E_s) = L(E'_{[s]})$ für alle $s \in S$. Zunächst beweisen wir die folgende Aussage:

> Für $w \in V_T^*$ gelte $sw \Longrightarrow_E^* \bar{s}$ und $[s]w \Longrightarrow_{E'}^* [s']$.
> Dann folgt $[\bar{s}] = [s']$.

Dies geschieht durch vollständige Induktion über die Länge des Wortes w. Für $|w| = 0$, also $w = \varepsilon$, sind die gegebenen Ableitungen trivial und können als

$$s = \bar{s},\ [s] = [s']$$

geschrieben werden. Folglich gilt $[\bar{s}] = [s']$. Es sei nun weiter $w = w'x$ mit $w' \in V_T^*, x \in V_T$. Wir betrachten die Ableitungen

$$sw'x \Longrightarrow^* \bar{s}x \Longrightarrow \bar{\bar{s}},\ [s]w'x \Longrightarrow^* [s']x \Longrightarrow [s''],$$

wobei laut Induktionsannahme $[\bar{s}] = [s']$ gilt. Die jeweils letzten Ableitungsschritte der beiden Ableitungen sind äquivalent zu

$$(\bar{s}x, \bar{\bar{s}}) \in F \text{ bzw. } ([s']x, [s'']) \in F'.$$

Wegen $[\bar{s}] = [s']$ und der Definition von F' folgt $([\bar{s}]x, [\bar{\bar{s}}]) \in F'$ mit $[\bar{\bar{s}}] = [s'']$. Damit ist der Induktionsbeweis abgeschlossen.

Es sei nun $w \in L(E_s)$ mit w und s wie in der obigen Aussage angegeben. Dann ist $\bar{s} \in S_1$. Wegen $[\bar{s}] = [s']$ und der Definition von S_1' gilt $[s'] \in S_1'$, und es folgt daher $w \in L(E'_{[s]})$. Umgekehrt gelte $w \in L(E'_{[s]})$. Dann ist $[s'] \in S_1'$. Wegen der Aussage ergibt sich $[s'] = [\bar{s}]$ und damit $[\bar{s}] \in S_1'$. Nach der Definition von S_1' existiert ein $r \in S_1$ mit $r \sim \bar{s}$. Wenn wir in Satz 1.4 $s' = r$, $s'' = \bar{s}$ und für das dortige w das leere Wort ε wählen, so erkennen wir sofort, daß $\bar{s} \in S_1$ gilt. Wir erhalten also $w \in L(E_s)$. Insgesamt ist dadurch $L(E_s) = L(E'_{[s]})$ bewiesen.

Nach Definition 1.14 muß noch gezeigt werden, daß die Zustände von E' paarweise inäquivalent sind. Es gelte $[s] \sim [s']$. Dann folgt nach Definition 1.12 und dem gerade geführten Beweis $L(E_s) = L(E'_{[s]}) = L(E'_{[s']}) = L(E_{s'})$. Wir erhalten also $s \sim s'$ und damit $[s] = [s']$. □

In Satz 1.5 wird lediglich die Existenz eines reduzierten Automaten bewiesen, der zu einem gegebenen Automaten äquivalent ist. Zur Konstruktion des reduzierten Automaten müssen die durch die Zustandsäquivalenz erzeugten Äquivalenzklassen explizit angegeben werden. Nach Satz 1.4 müßten dafür alle Wörter aus V_T^* betrachtet werden, also unendlich viele. Dies ist natürlich nicht möglich. Das Problem wird durch Satz 1.6 auf eine endliche Konstruktion reduziert.

Definition 1.15 Es sei E ein endlicher erkennender Automat und $m \in \mathbb{N}_0$. Dann heißen zwei Zustände s und $\bar{s}$ von E *m-äquivalent* (in Zeichen $s \sim_m \bar{s}$), wenn

$$\bigwedge_{\substack{w \in V_T^* \\ |w| \leq m}} \bigvee_{s', \bar{s}' \in S} ((sw \Longrightarrow^* s' \in S_1) \iff (\bar{s}w \Longrightarrow^* \bar{s}' \in S_1))$$

gilt. □

Die m-Äquivalenz ist offenbar eine Äquivalenzrelation.

Satz 1.6 Es sei E ein endlicher erkennender Automat mit $\text{card}(S) = n \geq 2$. Dann gilt

$$\bigwedge_{s, \bar{s} \in S} (s \sim_E \bar{s} \iff s \sim_{n-2} \bar{s}).$$

Beweis: Nach Definition 1.15 folgt für jedes $m \in \mathbb{N}$ aus der $(m+1)$-Äquivalenz die m-Äquivalenz, d.h., jede Äquivalenzklasse von $\sim_{m+1}$ ist enthalten in einer Äquivalenzklasse von $\sim_m$. Wenn e_m die Anzahl der Äquivalenzklassen von $\sim_m$ ist, dann gilt

$$(*) \qquad e_0 \leq e_1 \leq \ldots \leq e_m \leq \ldots \leq \text{card}(S).$$

Gilt $S_1 = \emptyset$ oder $S_1 = S$, so sind alle Zustände sowohl äquivalent als auch $(n-2)$-äquivalent.

Im folgenden sei also $S_1 \neq \emptyset$ und $S_1 \neq S$. Dann besitzt $\sim_0$ genau zwei Äquivalenzklassen, und zwar S_1 und $S - S_1$. Somit gilt $e_0 = 2$. Wegen $(*)$ existiert dann ein $m \in \mathbb{N}_0$ mit $m \leq \text{card}(S) - 2 = n - 2$ und $e_m = e_{m+1}$. Für dieses m folgt $\sim_m = \sim_{m+1}$. Nun gilt

$$
\begin{aligned}
s \sim_{m+1} \bar{s} &\iff \bigwedge_{\substack{w \in V_T^* \\ 0 \leq |w| \leq m+1}} \bigvee_{s', \bar{s}' \in S} ((sw \Longrightarrow^* s' \in S_1) \iff (\bar{s}w \Longrightarrow^* \bar{s}' \in S_1)) \\
&\iff s \sim_0 \bar{s} \wedge \bigwedge_{\substack{w \in V_T^* \\ 1 \leq |w| \leq m+1}} \bigvee_{s', \bar{s}' \in S} ((sw \Longrightarrow^* s' \in S_1) \iff (\bar{s}w \Longrightarrow^* \bar{s}' \in S_1)) \\
&\iff s \sim_0 \bar{s} \wedge \bigwedge_{x \in V_T} \bigwedge_{\substack{w' \in V_T^* \\ 0 \leq |w'| \leq m}} \bigvee_{s', \bar{s}' \in S} \\
&\qquad\qquad ((sxw' \Longrightarrow^* s' \in S_1) \iff (\bar{s}xw' \Longrightarrow^* \bar{s}' \in S_1)) \\
&\iff s \sim_0 \bar{s} \wedge \bigwedge_{x \in V_T} ((sx, s_1), (\bar{s}x, \bar{s}_1) \in F \Longrightarrow s_1 \sim_m \bar{s}_1).
\end{aligned}
$$

Wegen $\sim_m = \sim_{m+1}$ kann man in der letzten Zeile $\sim_m$ durch $\sim_{m+1}$ ersetzen. Durch entsprechende rückwärtige Verwendung der obigen logischen Äquivalenzen ergibt sich schließlich $s \sim_{m+2} \bar{s}$. Folglich gilt $\sim_{m+2} = \sim_{m+1}$ und entsprechend weiter $\sim_{m+\nu} = \sim_{m+1}$ für alle $\nu \in \mathbb{N}$ und damit schließlich $\sim_E = \sim_m$. □

Es sei E ein endlicher erkennender Automat. Es ergibt sich der folgende *Reduktionsalgorithmus* zur Bestimmung eines zu E reduzierten Automaten:

> Man konstruiere nach Satz 1.3 einen vereinfachten Automaten $\bar{E}$ von E. Die Zustände des reduzierten Automaten von $\bar{E}$, also die Klassen äquivalenter Zustände von $\bar{E}$, werden mit Hilfe von Satz 1.6 gewonnen, indem durch fortlaufende Verfeinerung die Äquivalenzklassen bezüglich $\sim_0$, $\sim_1$, usw. konstruiert werden, bis erstmalig $\sim_m = \sim_{m+1}$ für ein $m \in \mathbb{N}_0$ gilt. Die Klassen von $\sim_m$ bilden die Zustände des reduzierten Automaten. Spätestens für $m = n - 2$ kann das Verfahren abgebrochen werden. Bei der Verfeinerung ist zu beachten, daß für $s \sim_k \bar{s}$ mit $k \in \mathbb{N}_0$ nach den Überlegungen aus dem Beweis von Satz 1.6 die Relation $s \sim_{k+1} \bar{s}$ genau dann gilt, wenn
>
> $$\bigwedge_{x \in V_T} (((sx, s'), (\bar{s}x, \bar{s}') \in F) \implies (s' \sim_k \bar{s}'))$$
>
> erfüllt ist. Die weitere Konstruktion des Automaten erfolgt nach Satz 1.5.

Ohne Beweis geben wir den folgenden Satz an.

Satz 1.7 Es seien E und E' reduzierte endliche erkennende Automaten mit $L(E) = L(E')$. Dann sind E und E' isomorph, d.h. gleich bis auf die Bezeichnung der Zustände. □

Damit erhalten wir:

Satz 1.8 Es sei E ein endlicher erkennender Automat. Dann ist der reduzierte Automat E' von E bis auf Isomorphie der einzige endliche erkennende Automat mit $L(E') = L(E)$, dessen Anzahl von Zuständen minimal ist in bezug auf alle zu E äquivalenten Automaten. Er wird auch *Minimalautomat* genannt.

Beweis: Es sei $\bar{E}$ ein endlicher erkennender Automat mit $L(\bar{E}) = L(E)$. Zu zeigen ist $\text{card}(\bar{S}) \geq \text{card}(S')$. Wir konstruieren den reduzierten Automaten $\bar{E}'$ von $\bar{E}$ mit

$$L(\bar{E}') = L(\bar{E}) = L(E) = L(E').$$

Nach Satz 1.7 sind $\bar{E}'$ und E' isomorph, sie besitzen also die gleiche Anzahl von Zuständen. Es folgt $\text{card}(\bar{S}) \geq \text{card}(\bar{S}') = \text{card}(S')$. □

„Der" Minimalautomat ist mit Hilfe des Reduktionsalgorithmus in endlich vielen Schritten konstruierbar. Als einfache Folgerung ergibt sich der folgende Satz.

Satz 1.9 Es seien E und $\bar{E}$ endliche erkennende Automaten. Dann ist es entscheidbar (d.h., es existiert ein Algorithmus, der entscheidet), ob $L(E) = L(\bar{E})$ gilt oder nicht.

Beweis: Wir konstruieren die Minimalautomaten E' und $\bar{E}'$ von E bzw. $\bar{E}$. Durch Überprüfung, ob diese Automaten bis auf Isomorphie gleich sind, ergibt sich nach Satz 1.8, ob $L(E') = L(\bar{E}')$ gilt. Damit wird entschieden, ob $L(E) = L(\bar{E})$ erfüllt ist. □

1.4 Stochastische Akzeptoren

Wenn die nichtdeterministischen Zustandsüberführungen eines nichtdeterministischen endlichen erkennenden Automaten mit Wahrscheinlichkeiten versehen werden, kommen wir im wesentlichen zur Definition eines stochastischen Akzeptors. Zunächst führen wir den Begriff stochastische Matrix ein.

Definition 1.16 Es sei $\mathfrak{M} = (m_{i,j})$ eine $n \times n$-Matrix. Dann heißt $\mathfrak{M}$ *stochastische Matrix*, wenn das folgende gilt:

(a) $m_{i,j} \in \mathbb{R}$, $0 \leq m_{i,j} \leq 1$ für alle i, j, $1 \leq i, j \leq n$,

(b) $\sum_{j=1}^{n} m_{i,j} = 1$ für alle i, $i = 1, \ldots, n$. □

Da die Zeilensumme einer stochastischen Matrix 1 ist, kann jede ihrer Zeilen als eine Wahrscheinlichkeitsverteilung aufgefaßt werden.

Definition 1.17 $W = (A, \lambda)$ heißt *stochastischer Akzeptor*, wenn $A = (V, \mathfrak{M}, \pi, S_1)$ gilt, wobei folgende Eigenschaften erfüllt sind:

(a) Es gilt $V = S \cup V_T$ mit *Zustandsalphabet* $S = \{s_1, \ldots, s_n\}$ für ein $n \in \mathbb{N}$ und *Eingabealphabet* V_T, wobei $S \cap V_T = \emptyset$ ist.

(b) Wenn Π die Menge der Wahrscheinlichkeitsverteilungen über S ist, dann gilt $\pi = (\pi_1, \ldots, \pi_n) \in \Pi$, und π heißt *Anfangsverteilung*.

(c) $S_1 \subset S$ heißt *Menge der Endzustände*.

(d) $\mathfrak{M}$ ist eine Abbildung, die jedem $x \in V_T$ eine stochastische $n \times n$-Matrix $\mathfrak{M}(x)$ zuordnet.

(e) Es gilt $\lambda \in \mathbb{R}$, $0 \leq \lambda < 1$. λ heißt *Schnittpunkt* oder *Schwellwert* von W. □

Die Definition ist so zu interpretieren, daß sich der stochastische Akzeptor W zu Beginn mit der Wahrscheinlichkeit π_i im Anfangszustand s_i befindet. Wird im Zustand s_i das Symbol x eingegeben, so geht W mit Wahrscheinlichkeit $m_{i,j}(x)$ in den Folgezustand s_j über. $m_{i,j}(x)$ ist dabei das Element in der i-ten Zeile und j-ten Spalte von $\mathfrak{M}(x)$.

Definition 1.18 Es sei $W = (A, \lambda)$ ein stochastischer Akzeptor mit den Daten wie in Definition 1.17 angegeben.

(a) Die Abbildung $\mathfrak{M}$ werde zu einer Abbildung von V_T^* in die Menge der stochastischen $n \times n$-Matrizen durch

$$\mathfrak{M}(w) = \mathfrak{M}(a_1) \cdots \mathfrak{M}(a_r) \text{ für alle } w \in V_T^+ \text{ mit } w = a_1 \ldots a_r, a_i \in V_T \text{ und}$$
$$\mathfrak{M}(\varepsilon) = E_{n \times n}, \text{ wobei } E_{n \times n} \text{ die } n\text{-zeilige Einheitsmatrix ist,}$$

erweitert.

(b) Es werden Zahlen $\sigma_{1,i} \in \{0, 1\}$, $i = 1, \ldots, n$, durch $\sigma_{1,i} = 1$ für $s_i \in S_1$ und $\sigma_{1,i} = 0$ sonst gegeben. Damit wird ein Spaltenvektor

$$\bar{\sigma}_1 = \begin{pmatrix} \sigma_{1,1} \\ \vdots \\ \sigma_{1,n} \end{pmatrix}$$

definiert.

(c) Für $w \in V_T^*$ setzen wir $Pr(w) = \pi \cdot \mathfrak{M}(w) \cdot \bar{\sigma}_1$.

(d) $L(W) = \{w \mid w \in V_T^*,\ Pr(w) > \lambda\}$ heißt die *von W angenommene* oder *erkannte (akzeptierte) Sprache* (auch *stochastische Sprache* genannt). □

Das Element der i-ten Zeile und j-ten Spalte von $\mathfrak{M}(w)$ werde mit $m_{i,j}(w)$ bezeichnet. Man kann sich leicht überlegen, daß $m_{i,j}(w)$ die Wahrscheinlichkeit ist, mit der man aus dem Zustand s_i nach Abarbeitung des Wortes w in den Zustand s_j gelangt. Dann gibt $Pr(w)$ die Wahrscheinlichkeit an, mit der man, ausgehend von der Anfangsverteilung, nach Abarbeitung von w in einen Endzustand gelangt.

Satz 1.10 Jeder endliche erkennende Automat läßt sich als stochastischer Akzeptor $W = (A, \lambda)$ auffassen.

Beweis: Zunächst werde λ mit $0 \leq \lambda < 1$ beliebig gewählt. Es gelte $S = \{s_1, \ldots, s_n\}$, wobei s_1 der Anfangszustand von E ist. Für $A = (V, \mathfrak{M}, \pi, S_1)$ sind V und S_1 bereits durch die entsprechenden Daten von E bestimmt. Wir setzen $\pi = (1, 0, \ldots, 0)$. Weiter wird für jedes $x \in V_T$ die stochastische Matrix $\mathfrak{M}(x)$ dadurch definiert, daß in der i-ten Zeile genau dasjenige Element $m_{i,j}(x)$ den Wert 1 hat, für das $(s_i x, s_j) \in F$ in E gilt. Dies ist, da E deterministisch ist, bei festem i für genau ein j erfüllt. Alle anderen Elemente der i-ten Zeile werden auf 0 gesetzt. Offenbar sind die Arbeitsweisen von E und W in dem Sinne gleich, daß mit Wahrscheinlichkeit 1 der stochastische Akzeptor W für dieselben Eingaben dieselben Zustandüberführungen wie E durchführt. Ersichtlich gilt $L(E) = L(W)$. $\square$

Satz 1.11 Es sei

$$\mathfrak{M}(0) = \begin{pmatrix} 1 & 0 \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}, \quad \mathfrak{M}(1) = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ 0 & 1 \end{pmatrix}.$$

Dann hat für $k \in \mathbb{N}$, $\delta_i \in \{0, 1\}$, $i = 1, \ldots, k$, das Element q der stochastischen Matrix

$$\mathfrak{M}(\delta_1) \cdots \mathfrak{M}(\delta_k) = \begin{pmatrix} 1-q & q \\ 1-r & r \end{pmatrix}$$

die Dualbruchdarstellung $q = 0, \delta_k \delta_{k-1} \ldots \delta_1$.

Beweis: Der Beweis erfolgt durch vollständige Induktion über k. Für $k = 1$ erhalten wir für $\delta_1 = 0$ aufgrund der Definition von $\mathfrak{M}(0)$ den Wert $q = 0,0$, für $\delta_1 = 1$ aufgrund der Definition von $\mathfrak{M}(1)$ den Wert $q = \frac{1}{2} = 0,1$.

Wir nehmen nun an, daß die Aussage des Satzes für k erfüllt ist. Es sind wieder zwei Fälle zu betrachten. Es gilt

$$\mathfrak{M}(\delta_1) \cdots \mathfrak{M}(\delta_k) \cdot \mathfrak{M}(0) = \begin{pmatrix} 1-q & q \\ 1-r & r \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} = \begin{pmatrix} 1-q' & q' \\ 1-r' & r' \end{pmatrix}$$

mit $q' = \frac{1}{2} \cdot q = \frac{1}{2} \cdot 0, \delta_k \ldots \delta_1 = 0, 0\delta_k \ldots \delta_1$ sowie

$$\mathfrak{M}(\delta_1) \cdots \mathfrak{M}(\delta_k) \cdot \mathfrak{M}(1) = \begin{pmatrix} 1-q & q \\ 1-r & r \end{pmatrix} \cdot \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1-q'' & q'' \\ 1-r'' & r'' \end{pmatrix}$$

mit $q'' = \frac{1}{2} + \frac{q}{2} = 0,1 + \frac{1}{2} \cdot 0, \delta_k \ldots \delta_1 = 0,1 + 0, 0\delta_k \ldots \delta_1 = 0, 1\delta_k \ldots \delta_1$. $\square$

Der nächste Satz zeigt, daß es Sprachen gibt, die zwar von stochastischen Akzeptoren, nicht aber von endlichen erkennenden Automaten erkannt werden.

Satz 1.12 Es gelte $A = (V, \mathfrak{M}, \pi, S_1)$ mit $S = \{s_0, s_1\}$, $V_T = \{0, 1\}$, $\pi = (1, 0)$, $S_1 = \{s_1\}$ und $\mathfrak{M}(0)$ und $\mathfrak{M}(1)$ wie in Satz 1.11. Dann existiert eine Zahl $\lambda \in \mathbb{R}$, $0 \leq \lambda < 1$, für die kein endlicher erkennender Automat E mit $L((A, \lambda)) = L(E)$ existiert.

Beweis: Für beliebige $k \in \mathbb{N}$ und $\delta_i \in \{0,1\}$ betrachten wir die Wörter $w = \delta_1 \ldots \delta_k$. Nach Satz 1.11 gilt

$$Pr(w) = (1,0) \cdot \mathfrak{M}(w) \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} = 0, \delta_k \ldots \delta_1.$$

Alle diese Werte von $Pr(w)$ liegen dicht im Intervall $[0,1]$, d.h., in jedem beliebigen Teilintervall befindet sich ein solcher Wert. Für $\lambda, \lambda_1 \in \mathbb{R}$, $0 \leq \lambda < \lambda_1 < 1$, existiert folglich ein $w \in \{0,1\}^+$ mit $\lambda < Pr(w) < \lambda_1$. Setzen wir $W = (A, \lambda)$ und $W_1 = (A, \lambda_1)$, so gilt nach Definition 1.18.(d) $w \in L(W)$ und $w \notin L(W_1)$. Für jedes solche Zahlenpaar (λ, λ_1) gilt also $L(W_1) \subsetneqq L(W)$. Wir erkennen damit, daß die Sprachen $L(W)$ für stochastische Akzeptoren $W = (A, \lambda)$ mit $\lambda \in \mathbb{R}$, $0 \leq \lambda < 1$, eine überabzählbare Familie von paarweise verschiedenen Sprachen liefern.

Andererseits gibt es jedoch bis auf Isomorphie (bis auf die Bezeichnung der Zustände) nur abzählbar viele endliche erkennende Automaten mit Eingabealphabet $\{0,1\}$. Das ergibt sich daraus, daß es zu jeder Zustandsanzahl nur endlich viele solche Automaten gibt. □

Zusammen mit Satz 1.10 erkennen wir, daß die Familie der durch endliche erkennende Automaten akzeptierten Sprachen echt in der Familie der durch stochastische Akzeptoren erkannten Sprachen enthalten ist.

Kapitel 2

Sprachen und Grammatiken

2.1 Einführung und Definitionen

In Kapitel 1 wurden endliche erkennende Automaten E als Erkennungseinrichtungen für Sprachen $L = L(E)$ betrachtet. Das bedeutet auch, daß sie die *charakteristische Funktion* dieser Sprachen berechnen, nämlich

$$\chi_L(w) = \begin{cases} 0, & \text{falls } w \notin L \\ 1, & \text{falls } w \in L \end{cases} .$$

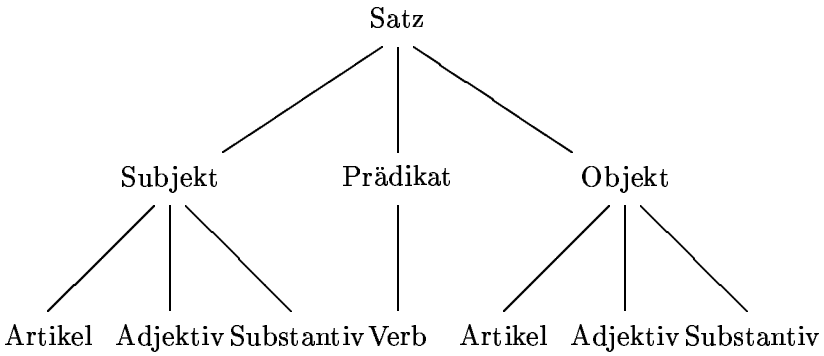
Wenn eine Sprache L endlich viele Elemente hat, dann ist sie durch Aufzählung ihrer Elemente darstellbar. Anderenfalls ist eine Formulierung nötig, die eine endliche Beschreibung der Sprache ermöglicht. Dies kann z.B. wie in Kapitel 1 durch einen endlichen erkennenden Automaten geschehen. Wir werden in Kapitel 3 weitere Typen von Automaten (Akzeptoren) einführen, die verschiedene Klassen von Sprachen mit Hilfe von Endzuständen erkennen. Neben Erkennungs- sind aber auch Generierungsverfahren von Bedeutung, die nach gewissen Regeln Wörter der Sprache erzeugen. Besonders wichtig sind dabei Grammatiken, die wir in diesem Kapitel ausführlich behandeln. Für verschiedene Programmiersprachen kann die Syntax durch eine Grammatik beschrieben werden. In [30], Kapitel 8, ist dies z.B. für eine spezielle Programmiersprache durchgeführt worden. Zunächst geben wir einen Versuch an, die deutsche Sprache durch eine Grammatik zu erzeugen.

Beispiel 2.1 Ein Ausschnitt einer möglichen Grammatik für die deutsche Sprache ist

Satz → Subjekt Prädikat	⋮
Satz → Subjekt Prädikat Objekt,	Objekt → Artikel Adjektiv Substantiv,
Subjekt → Substantiv,	⋮
Subjekt → Artikel Substantiv,	Prädikat → Verb,
Subjekt → Artikel Adjektiv Substantiv,	Prädikat → Hilfsverb Verb.

Diese Regeln müssen noch erweitert werden, da Artikel, Substantiv, Adjektiv, Verb und Hilfsverb durch konkrete derartige Wörter ersetzt werden müssen, z.B. durch eine

Regel „Verb → bauen“. Eine Ableitung nach dieser Grammatik kann durch einen *Strukturbaum* dargestellt werden, etwa



Durch Einsetzen geeigneter Wörter, d.h. durch zusätzliche Verwendung von Regeln, die die Einträge des Duden liefern, erhält man etwa den Satz „Die fleißigen Studenten hören eine gute Vorlesung“. Das Problem ist dabei jedoch, daß auch unsinnige Sätze gebildet werden können. Eine natürliche Sprache kann also durch eine formale Grammatik nicht befriedigend dargestellt werden. □

Wir stellen nun verschiedene Möglichkeiten zur Festlegung von Sprachen $L \subset V^*$ zusammen:

(a) Auflistung der Elemente: Dies ist nur für endliche Sprachen möglich.
(b) Erzeugungs- oder Generierungsverfahren: Es werden Regeln angegeben, nach denen gewisse Elemente aus V^* erzeugt werden können. Es wird festgelegt, daß L genau die Menge der Wörter ist, die durch diese Regeln erzeugt werden.
(c) Erkennungs- oder Rekognitionsverfahren: Es wird ein Algorithmus angegeben, der bei Anwendung auf ein $w \in V^*$ feststellt, ob $w \in L$ gilt oder nicht. Dies haben wir in Kapitel 1 bereits kennengelernt. Es kann sich aber auch um ein Verfahren handeln, das bei Anwendung auf $w \in L$ nach endlich vielen Schritten abbricht mit der Aussage, daß $w \in L$ ist, jedoch für $w \notin L$ nicht abbricht.
(d) Konstruktionsverfahren: Es wird ein Ausdruck angegeben, der L darstellt.
(e) Lösung einer Gleichung: Es wird eine Gleichung über V^* angegeben. Die Lösungen dieser Gleichungen bestimmen eine (oder mehrere) formale Sprachen (siehe z.B. [30], Kapitel 6).

In Definition 1.4 haben wir bereits den Begriff eines kombinatorischen Systems (V, F) eingeführt. Der Ableitungsbegriff bei kombinatorischen Systemen wurde in Definition 1.5 angegeben. Durch Auszeichnung von gewissen Teilmengen von V^* haben wir endliche erkennende Automaten definiert. Durch andere Auszeichnungen erhalten wir allgemeine Erzeugungs- und Erkennungssysteme.

Definition 2.1 Es sei $KS = (V, F)$ ein kombinatorisches System, und es sei $AX \subset V^*$ (*Axiomenmenge* genannt). Dann heißt

(a) $L_g(KS, AX) = \{Q \mid P \Longrightarrow^* Q,\ P \in AX\}$ die *von* (KS, AX) *erzeugte Sprache* und

(b) $L_a(KS, AX) = \{P \mid P \Longrightarrow^* Q,\ Q \in AX\}$ die *von* (KS, AX) *erkannte* oder *akzeptierte Sprache.* □

Im folgenden wird V unterteilt in ein Terminalalphabet V_T und ein Nichtterminalalphabet V_N. Zur Sprache sollen nur noch die Wörter aus V_T^* gehören. Dies verdeutlichen wir zunächst an einem Beispiel.

Beispiel 2.2 (a) Es sei $KS = (\{a, b, X\}, \{X \to \varepsilon, X \to aXb\})$. Wir wählen $V_N = \{X\}$ und $V_T = \{a, b\}$. Dann ist $L_g(KS, \{X\}) \cap \{a, b\}^* = \{a^i b^i \mid i \in \mathbb{N}_0\} = \{\varepsilon, ab, a^2b^2, a^3b^3, \ldots\}$.

(b) Es sei $KS = (\{a, b\}, \{ab \to \varepsilon\})$. Wir wählen $V_T = V = \{a, b\}$. Dann ist $L' = L_a(KS, \{\varepsilon\})$ die Menge der Wörter über $\{a, b\}$, für die eine Ersetzung von Ketten ab durch ε schließlich ε ergibt. So gilt etwa $a^n b^n \in L'$, $a^i b^i a^j b^j \in L'$, jedoch z.B. $ab^n a^n b \notin L'$ für $n > 1$. □

Als Verallgemeinerung dieses Beispiels kommen wir zu Grammatiken. Bevor wir jedoch darauf eingehen, definieren wir noch eine scheinbar eingeschränkte Form eines Ersetzungssystems, das wir allerdings im weiteren Verlauf der Vorlesung nicht weiter betrachten werden.

Definition 2.2 Es sei $KS = (V, F)$ ein kombinatorisches System.

(a) $KS = (V, F)$ heißt *Markovscher Normalalgorithmus*, wenn folgendes gilt:

(α) Für die Elemente von F ist eine lineare Ordnung gegeben. Falls $\mathrm{card}(F) = k$, $k \in \mathbb{N}$, gilt, so sei die Ordnung durch die Reihenfolge

$$(P_1, Q_1), \ldots, (P_k, Q_k)$$

definiert.

(β) Eine *Menge von Endproduktionen* $F_1 \subset F$ werde ausgezeichnet. Ihre Elemente werden auch speziell in der Form $(P, .Q)$ geschrieben.

(b) Es seien $P, Q \in V^*$. P *erzeugt direkt* Q (in Zeichen $P \Longrightarrow Q$), wenn die folgenden Eigenschaften erfüllt sind:

(α) Es existieren $i \in \mathbb{N}$, $1 \le i \le k$, $P', P'' \in V^*$ mit $P = P'P_iP''$, $Q = P'Q_iP''$.

(β) Keines der Wörter P_j mit $j < i$ ist ein Teilwort von P.

(γ) P_i ist genau einmal ein Teilwort von $P'P_i$.

(δ) Es gilt $(P_i, Q_i) \notin F_1$, und eines der Wörter $P_1, \ldots, P_k$ ist ein Teilwort von Q.

(c) P *erzeugt zum Ende des Ersetzungsprozesses direkt* Q (in Zeichen $P \Longrightarrow. Q$), wenn die Bedingungen (b)(α) bis (b)(γ) erfüllt sind, (b)(δ) jedoch nicht (d.h.: es wird eine Endproduktion angewendet, oder es ist keine Produktion mehr anwendbar).

(d) P *erzeugt* Q (in Zeichen $P \Longrightarrow^*. Q$), wenn Wörter $P = R_0, R_1, \ldots, R_u, Q \in V^*$ für ein $u \in \mathbb{N}_0$ existieren, so daß $R_j \Longrightarrow R_{j+1}$ für alle j, $0 \le j \le u - 1$, gilt und außerdem $R_u \Longrightarrow. Q$ erfüllt ist, oder anderenfalls ist $P = Q$, und kein $P_1, \ldots, P_k$ ist Teilwort von P. □

Bei jedem Schritt des Ersetzungsprozesses wird die bezüglich der linearen Ordnung erste anwendbare Produktion gesucht (siehe (b)(β)). Dann wird das am weitesten links stehende Teilwort des vorliegenden Wortes, das der linken Seite dieser Produktion entspricht, ersetzt (siehe (b)(γ)). Damit wird erreicht, daß der Markovsche Normalalgorithmus in einem Schritt höchstens ein Wort ableiten kann. Der Ersetzungsprozeß endet mit der Anwendung einer Produktion aus F_1 oder wenn keine Produktion aus F mehr anwendbar ist (siehe (c)). Wir sehen also, daß es für ein beliebiges $P \in V^*$ höchstens ein $Q \in V^*$ gibt mit $P \Longrightarrow_{\cdot}^{*} Q$. Falls ein solches Q existiert, sagen wir, daß der Markovsche Normalalgorithmus mit P *hält* und das Wort P in Q *übersetzt*. Anderenfalls läuft er mit P in einer *Schleife*.

Beispiel 2.3 Es sei $KS = (\{a,b\}, \{(a,\varepsilon), (b,\varepsilon), (\varepsilon, {}_{\cdot}aba)\})$ ein Markovscher Normalalgorithmus. Die lineare Ordnung sei durch die Reihenfolge der Produktionen gegeben. Ein beliebiges Wort $P \in \{a,b\}^*$ wird in aba übersetzt. Wir erhalten z.B.

$$aabaa \Longrightarrow abaa \Longrightarrow baa \Longrightarrow ba \Longrightarrow b \Longrightarrow \varepsilon \Longrightarrow_{\cdot} aba. \quad \square$$

Im weiteren Verlauf diese Kapitels wollen wir uns mit Grammatiken und den von ihnen erzeugten Sprachen beschäftigen.

Definition 2.3 $G = (V_N, V_T, X_0, F)$ heißt *(erzeugende) Grammatik*, wenn
(a) V_N und V_T Alphabete sind mit $V_N \cap V_T = \emptyset$ (V_T heißt Menge der *Endsymbole* oder *Terminalzeichen* und V_N Menge der *Nichtendsymbole* oder *Nichtterminalzeichen*),
(b) $X_0 \in V_N$ ist (X_0 heißt *Anfangssymbol*) und
(c) F eine endliche Menge von Paaren (P,Q) ist mit $Q \in V^*$, $V = V_N \cup V_T$, und $P \in V^*V_NV^*$ ($(P,Q) \in F$ heißt *Ersetzungsregel* oder *Produktion*). $\square$

Falls nichts anderes gesagt wird, vereinbaren wir $V = V_N \cup V_T$. Offenbar bestimmt eine Grammatik G das durch G induzierte kombinatorische System (V, F). Folglich läßt sich Definition 1.5 auch für Grammatiken anwenden. In G bedeuten also „erzeugt (direkt)“, „Ableitung“, „$\Longrightarrow$“ und „$\Longrightarrow^*$“ dasselbe wie im induzierten kombinatorischen System.

Definition 2.4 Es sei $G = (V_N, V_T, X_0, F)$ eine erzeugende Grammatik. Dann heißt

$$L(G) = \{w \mid w \in V_T^*, X_0 \Longrightarrow^* w\}$$

die *von G erzeugte Sprache*. $\square$

Als duales Konzept einer erzeugenden Grammatik erhalten wir das einer analysierenden Grammatik.

Definition 2.3* $G = (V_N, V_T, X_0, F)$ heißt *analysierende Grammatik*, wenn
(a) wie in Definition 2.3,
(b) wie in Definition 2.3 gegeben ist und
(c) F eine endliche Menge von Paaren (P,Q) ist mit $P \in V^*$ und $Q \in V^*V_NV^*$. $\square$

Dual zu Definition 2.4 erhalten wir

Definition 2.4* Es sei $G = (V_N, V_T, X_0, F)$ eine analysierende Grammatik. Dann heißt

$$L(G) = \{w \mid w \in V_T^*, w \Longrightarrow^* X_0\}$$

die *von G erkannte Sprache.* □

Definition 2.5 Es seien G und G' erzeugende oder analysierende Grammatiken. G und G' heißen *äquivalent*, wenn $L(G) = L(G')$ gilt. □

Wenn wir im folgenden von einer „Grammatik“ sprechen, dann ist immer eine erzeugende Grammatik gemeint.

Beispiel 2.4 Wir betrachten die Sprache $L = \{a^i b^i \mid i \in \mathbb{N}_0\}$. Sie wird von der Grammatik

$$(\{X\}, \{a, b\}, X, \{X \to \varepsilon, X \to aXb\})$$

erzeugt und von der analysierenden Grammatik

$$(\{X\}, \{a, b\}, X, \{\varepsilon \to X, aXb \to X\})$$

erkannt. □

Man beachte, daß das leere Wort ε als linke Seite nur bei analysierenden Grammatiken erlaubt ist, nicht jedoch bei erzeugenden Grammatiken. Das Dualitätsprinzip dieses Beispiels ist allgemein anwendbar. Dies zeigt der folgende Satz.

Satz 2.1 Es sei $G = (V_N, V_T, X_0, F)$ eine erzeugende oder analysierende Grammatik. Es werde eine weitere Produktionenmenge F' durch $(Q, P) \in F' \iff (P, Q) \in F$ und damit eine analysierende bzw. erzeugende Grammatik $G' = (V_N, V_T, X_0, F')$ definiert. Dann gilt $L(G) = L(G')$.

Beweis: Ohne Beschränkung der Allgemeinheit sei G eine erzeugende Grammatik. Dann ist G' eine analysierende Grammatik. Zu zeigen ist, daß es für alle $w \in V_T^*$ die Ableitung $X_0 \Longrightarrow_G^* w$ genau dann gibt, wenn $w \Longrightarrow_{G'}^* X_0$ existiert. Wir beweisen zunächst durch Induktion über die Ableitungslänge k für Wörter $P_0, \ldots, P_k \in V^*$ die Gültigkeit von

$$(*) \quad \bigwedge_{k \in \mathbb{N}} ((P_0 \Longrightarrow_G P_1 \Longrightarrow_G \ldots \Longrightarrow_G P_k) \iff (P_k \Longrightarrow_{G'} P_{k-1} \Longrightarrow_{G'} \ldots \Longrightarrow_{G'} P_0)).$$

Nach Definition von F' ist diese Aussage für $k = 1$ erfüllt. Nach Induktionsannahme sei sie für $k \geq 1$ erfüllt. Dann gilt sowohl $(*)$ als auch, und zwar nach Definition von F', $(P_k \Longrightarrow_G P_{k+1}) \iff (P_{k+1} \Longrightarrow_{G'} P_k)$. Damit erhalten wir die Aussage für $k + 1$.

Als Spezialfall ergibt sich $(X_0 \Longrightarrow_G^* w) \iff (w \Longrightarrow_{G'}^* X_0)$ und damit $L(G) = L(G')$. □

Wir nennen G' die zu G *duale Grammatik.*

Beispiel 2.5 Im allgemeinen ist es schwierig nachzuweisen, daß eine Grammatik eine bestimmte Wortmenge *und nur diese* erzeugt. Man betrachte

$$G = (\{X_0, X_1, X_2\}, \{a, b, c\}, X_0, \{X_0 \to abc,\ X_0 \to aX_1bc,\ X_1b \to bX_1,$$

$$X_1c \to X_2bcc,\ bX_2 \to X_2b,\ aX_2 \to aaX_1,\ aX_2 \to aa\}).$$

Wir behaupten $L(G) = \{a^ib^ic^i \mid i \in \mathbb{N}\}$. Die Produktion $X_0 \to abc$ liefert das Wort $a^1b^1c^1 \in L(G)$. Mit $X_0 \to aX_1bc$ gelangt man zu $a^1X_1b^1c^1$. Allgemein muß man für ein bereits erzeugtes Wort $a^iX_1b^ic^i, i \in \mathbb{N}$, zunächst genau i-mal die dritte Produktion, einmal die vierte Produktion und i-mal die fünfte Produktion anwenden:

$$a^iX_1b^ic^i \Longrightarrow^i a^ib^iX_1c^i \Longrightarrow a^ib^iX_2bc^{i+1} \Longrightarrow^i a^iX_2b^{i+1}c^{i+1}.$$

Dabei haben wir die Anzahl der direkten Ableitungsschritte als oberen Index an $\Longrightarrow$ herangeschrieben. Andere Ableitungen sind für $a^iX_1b^ic^i$ nicht möglich. Dann hat man die Wahl zwischen der sechsten und siebten Produktion, also

$$a^iX_2b^{i+1}c^{i+1} \Longrightarrow a^{i+1}X_1b^{i+1}c^{i+1} \quad \text{oder}$$

$$a^iX_2b^{i+1}c^{i+1} \Longrightarrow a^{i+1}b^{i+1}c^{i+1}.$$

Folglich erhalten wir $L(G)$ wie angegeben. □

Im folgenden teilen wir Grammatiken in verschiedene Klassen ein.

Definition 2.6 Es sei $G = (V_N, V_T, X_0, F)$ eine Grammatik. G ist *vom Typ* i, $i = 0, 1, 2, 3$, wenn die Einschränkung (i) für F erfüllt ist:

(0) Keine Einschränkung.

(1) Jede Produktion in F ist von der Form $Q_1XQ_2 \to Q_1PQ_2$ mit $Q_1, Q_2 \in V^*$, $X \in V_N$ und $P \in V^+$ mit der eventuellen Ausnahme von $X_0 \to \varepsilon$. Im Ausnahmefall darf X_0 nicht auf der rechten Seite einer Produktion von F vorkommen.

(2) Jede Produktion in F ist von der Form $X \to P$ mit $X \in V_N$ und $P \in V^*$.

(3) Jede Produktion in F ist von einer der Formen $X \to Yw$ oder $X \to w$ mit $X, Y \in V_N$ und $w \in V_T^*$.

Es sei $i \in \{0, 1, 2, 3\}$ und L eine Sprache. L ist *vom Typ* i, wenn $L = L(G)$ gilt, wobei G vom Typ i ist. $\mathfrak{L}_i = \{L \mid L = L(G),\ G \text{ vom Typ } i\}$ heißt *Familie der Sprachen vom Typ* i. □

Eine Grammatik vom Typ 1 heißt *kontextsensitiv*, da das Symbol X aus der Definition nur ersetzt werden kann, wenn es im Kontext $Q_1 \ldots Q_2$ steht. Grammatiken vom Typ 2 heißen auch *kontextfrei*, da das Zeichen X unabhängig vom Kontext ersetzt wird. Eine Grammatik vom Typ 3 wird *regulär* genannt. Die erzeugten Sprachen werden entsprechend als *kontextsensitive*, *kontextfreie* bzw. *reguläre* Sprachen bezeichnet.

Definition 2.7 Es sei G eine Grammatik. G heißt *monoton*, wenn für alle Produktionen $(P, Q) \in F$ die Ungleichung $|P| \leq |Q|$ gilt mit der eventuellen Ausnahme von (X_0, ε). Dann darf X_0 jedoch nicht auf der rechten Seite einer Produktion vorkommen. □

Offenbar ist jede Grammatik vom Typ 1 auch monoton. Umgekehrt werden wir in Satz 2.7 zeigen, daß jede Sprache, die von einer monotonen Grammatik erzeugt wird, vom Typ 1 ist. Die Grammatik aus dem Beispiel 2.5 ist vom Typ 0 und monoton. Die erzeugte Sprache ist also vom Typ 1. Die Sprache aus Beispiel 2.4 ist vom Typ 2. Wir geben nun noch ein Beispiel für eine Typ-3-Grammatik an.

Beispiel 2.6 Gegeben sei die Typ-3-Grammatik

$$G = (\{X_0, X_1\}, \{a, b\}, X_0, \{X_0 \to X_0 b,\ X_0 \to X_1 b,\ X_1 \to X_1 a,\ X_1 \to a\}).$$

Die von G erzeugte Sprache ist

$$L(G) = \{a^i b^k \mid i, k \in \mathbb{N}\} = \{a\}^+\{b\}^+,$$

denn es sind nur Ableitungen der Form

$$X_0 \Longrightarrow^{k-1} X_0 b^{k-1} \Longrightarrow X_1 b^k \Longrightarrow^{i-1} X_1 a^{i-1} b^k \Longrightarrow a^i b^k$$

möglich. Für ein Alphabet V_T erhalten wir die Sprache $L(G') = V_T^*$ durch die Typ-3-Grammatik $G' = (\{X_0\}, V_T, X_0, \{X_0 \to X_0 a, X_0 \to \varepsilon \mid a \in V_T\})$. □

2.2 Operationen bei Sprachen

Definition 2.8 Es sei $\mathfrak{L}$ eine Familie von Sprachen, und für $n \in \mathbb{N}$ sei ω eine n-stellige Operation auf $\mathfrak{L}$. $\mathfrak{L}$ heißt *abgeschlossen unter* ω, wenn $\omega(L_1, \ldots, L_n) \in \mathfrak{L}$ für alle $L_1, \ldots, L_n \in \mathfrak{L}$ gilt. □

Wir stellen zunächst die wichtigsten, dem Leser sicherlich bereits bekannten, Sprachoperationen zusammen.

Bezeichnungen Es seien L, L_1 und L_2 Sprachen über V, V_1 bzw. V_2. Wir betrachten die folgende Tabelle:

Vereinigung	$L_1 \cup L_2 = \{w \mid w \in L_1 \vee w \in L_2\}$
Durchschnitt	$L_1 \cap L_2 = \{w \mid w \in L_1 \wedge w \in L_2\}$
Differenz	$L_1 - L_2 = \{w \mid w \in L_1 \wedge w \notin L_2\}$
Komplement	$\complement L = V^* - L$
Konkatenation, Produkt	$L_1 L_2 = \{w_1 w_2 \mid w_1 \in L_1 \wedge w_2 \in L_2\}$
Iteration	$L^* = \bigcup_{i=0}^{\infty} L^i$ L^i : i-fache Konkatenation von L mit sich selbst
ε-freie Iteration	$L^+ = \bigcup_{i=1}^{\infty} L^i$

Definition 2.9 Es seien V und V' Alphabete. Eine Abbildung $h : V^* \to V'^*$ mit

$$h(\varepsilon) = \varepsilon, \bigwedge_{w,w' \in V^*} h(ww') = h(w)h(w')$$

heißt *Homomorphismus* von V^* in V'^*. □

Offensichtlich ist ein solcher Homomorphismus h eindeutig bestimmt, wenn für jedes $a \in V$ der Bildwert $h(a)$ festgelegt ist. Eine Erweiterung des Begriffs Homomorphismus erhalten wir durch den einer Substitution.

Definition 2.10 Es seien V und V' Alphabete. Eine *Substitution* σ von V^* in V'^* ist durch eine Abbildung $\sigma : V^* \to \mathfrak{P}(V'^*)$ mit

$$\sigma(\varepsilon) = \{\varepsilon\}, \bigwedge_{w,w' \in V^*} \sigma(ww') = \sigma(w)\sigma(w')$$

definiert. Für eine Sprache $L \subset V^*$ werde

$$\sigma(L) = \{v \mid v \in \sigma(w),\ w \in L\} \left(= \bigcup_{w \in L} \sigma(w) \right)$$

gesetzt. □

Eine Substitution σ ist eindeutig bestimmt, wenn für jedes $a \in V$ die Menge $\sigma(a) \subset V'^*$ festgelegt ist. $\sigma(a)$ ist also eine Sprache über V'. Ein Homomorphismus h kann als eine spezielle Substitution σ mit $\sigma(a) = \{h(a)\}$ für alle $a \in V$ aufgefaßt werden.

Definition 2.11 Es sei σ eine Substitution von V^* in V'^*. σ heißt *reguläre Substitution*, wenn $\sigma(a)$ für alle $a \in V$ regulär ist. σ heißt *ε-freie Substitution*, wenn $\varepsilon \notin \sigma(a)$ für alle $a \in V$ gilt. □

Ein Homomorphismus h als spezielle Substitution ist nach Definition 2.11 genau dann ε-frei, wenn $h(a) \neq \varepsilon$ für alle $a \in V$ gilt.

Definition 2.12 Es sei $\mathfrak{L}$ eine Familie von Sprachen und σ eine Substitution von V^* in V'^*. $\mathfrak{L}$ heißt *abgeschlossen unter* σ, wenn $\sigma(a) \in \mathfrak{L}$ für alle $a \in V$ gilt und aus $L \in \mathfrak{L}$ die Beziehung $\sigma(L) \in \mathfrak{L}$ folgt. □

Satz 2.2 Es sei $G = (V_N, V_T, X_0, F)$ eine Grammatik. Dann ist eine äquivalente Grammatik $G' = (V'_N, V_T, X_0, F')$ konstruierbar, die die Eigenschaft besitzt, daß jede Produktion aus F', die Elemente aus V_T enthält, von der Form $X \to a$ mit $X \in V'_N$ und $a \in V_T$ ist. Ist dabei G vom Typ 0, 1, 2 oder monoton, so ist auch G' vom Typ 0, 1, 2 oder monoton.

Beweis: Für jedes $a \in V_T$ führen wir ein neues Nichtterminalzeichen A_a ein und setzen

$$V'_N = V_N \cup \{A_a \mid a \in V_T\}.$$

Dabei wollen wir, wie auch in Zukunft, vereinbaren, daß jedes neue Zeichen, das bei einer Konstruktion eingeführt wird, von den vorher eingeführten Zeichen verschieden ist. Wir haben es also oben mit einer disjunkten Vereinigung zu tun.

Die Produktionenmenge F' von G' ergibt sich durch Ersetzung aller Symbole $a \in V_T$ in allen Produktionen von F durch A_a und anschließende Ergänzung durch Produktionen

$$(*) \qquad A_a \to a \text{ für alle } a \in V_T.$$

Offenbar hat dann F' die gewünschte Gestalt, und der zweite Teil des Satzes ist erfüllt (siehe Definition 2.6).

Zu zeigen ist $L(G) = L(G')$. Ist $w = a_1 \dots a_n \in L(G)$, $a_1, \dots, a_n \in V_T$, $n \in \mathbb{N}$, dann konstruieren wir zu jeder Ableitung von w in G eine in G', indem wir zunächst jedes $a \in V_T$ durch A_a ersetzen. Damit läßt sich in G' das Wort $A_{a_1} \dots A_{a_n}$ erzeugen. Unter zusätzlicher Verwendung der Produktionen $(*)$ ist dann w gemäß G' erzeugbar. Ist $\varepsilon \in L(G)$, so wird ohne abschließende Anwendung von Produktionen $(*)$ das leere Wort $\varepsilon \in L(G')$ abgeleitet. Damit ist $L(G) \subset L(G')$ bewiesen.

Für die umgekehrte Richtung definieren wir einen Homomorphismus $h : (V'_N \cup V_T)^* \to (V_N \cup V_T)^*$ mit

$$h(A_a) = a \text{ für } a \in V_T \text{ und } h(x) = x \text{ für } x \in V_N \cup V_T.$$

Wir nehmen nun an, daß $P \Longrightarrow_{G'} Q$ gilt. Wird in diesem Ableitungsschritt eine Produktion $(*)$ benutzt, so ist $h(P) = h(Q)$. Anderenfalls gilt $h(P) \Longrightarrow_G h(Q)$. Insgesamt folgt $h(P) \Longrightarrow_G^* h(Q)$. Für $w \in L(G')$ erhalten wir somit die Ableitung $X_0 = h(X_0) \Longrightarrow_G^* h(w) = w$. Daraus ergibt sich $w \in L(G)$. $\square$

Satz 2.3 Jede der Familien $\mathfrak{L}_i$, $i = 0, 1, 2, 3$, enthält alle endlichen Sprachen und ist abgeschlossen unter den Operationen Vereinigung, Konkatenation und Iteration.

Beweis: Es sei $n \in \mathbb{N}$ und V_T ein Alphabet. $L = \{w_1, \dots w_n\} \subset V_T^*$ sei eine endliche Sprache. L wird von der Grammatik

$$(\{X\}, V_T, X, \{X \to w_1, \dots, X \to w_n\})$$

und die leere Sprache $\emptyset$ durch die Grammatik

$$(\{X\}, V_T, X, \{X \to X\})$$

erzeugt. Beide Grammatiken sind vom Typ 0, 1, 2 und auch 3.

Im folgenden seien $G = (V_N, V_T, X_0, F)$ und $G' = (V'_N, V'_T, X'_0, F')$ Grammatiken vom Typ i, $i = 0, 1, 2, 3$, und wir setzen $L(G) = L$, $L(G') = L'$. Ohne Beschränkung der Allgemeinheit gelte $V_N \cap V'_N = \emptyset$. Außerdem nehmen wir an, daß auf den linken Seiten der Produktionen von F und F' keine Terminalsymbole vorkommen. Die letzte

Bedingung ist für Grammatiken des Typs 2 oder 3 nach Definition 2.6 erfüllt. Für Grammatiken vom Typ 0 oder 1 ist diese Forderung nach Satz 2.2 erfüllbar.

Als erstes betrachten wir die Operation der Vereinigung. Zunächst sei $i \neq 1$. Wegen $V_N \cap V'_N = \emptyset$ wird dann die Sprache $L \cup L'$ offenbar von der Typ-i-Grammatik

$$(V_N \cup V'_N \cup \{Y_0\}, V_T \cup V'_T, Y_0, F \cup F' \cup \{Y_0 \to X_0, Y_0 \to X'_0\})$$

erzeugt. Für $i = 1$ und $\varepsilon \notin L$, $\varepsilon \notin L'$ kann direkt dieselbe Konstruktion durchgeführt werden. Ist jedoch $\varepsilon \in L \cup L'$, dann werden zunächst die Produktionen $X_0 \to \varepsilon$ und $X'_0 \to \varepsilon$ (falls vorhanden) entfernt, dann die obige Konstruktion durchgeführt und anschließend $Y_0 \to \varepsilon$ zur Produktionenmenge hinzugefügt.

Wir kommen zur Konkatenation. Für $i = 3$ betrachten wir die Produktionen

$$X \to w,\ X \in V'_N,\ w \in V'^*_T$$

aus F'. Bei einer Ableitung gemäß G' können sie nur im letzten Ableitungsschritt angewendet werden. Wir ersetzen diese Produktionen daher durch die Produktionen $X \to X_0 w$ und erhalten so eine neue Produktionenmenge F_1. Dann ist $G_1 = (V_N \cup V'_N, V_T \cup V'_T, X'_0, F_1 \cup F)$ eine Typ-3-Grammatik mit $L(G_1) = LL'$.

Es sei weiter $i = 2$, $i = 0$ oder $i = 1$, wobei für $i = 1$ zusätzlich angenommen wird, daß $\varepsilon \notin L \cup L'$ gilt. Dann definieren wir die Typ-i-Grammatik

$$G_1 = (V_N \cup V'_N \cup \{Y_0\}, V_T \cup V'_T, Y_0, F \cup F' \cup \{Y_0 \to X_0 X'_0\}).$$

Jedes Wort aus LL' kann gemäß G_1 erzeugt werden. Umgekehrt sei

$$Y_0 \Longrightarrow Q_1 \Longrightarrow \ldots \Longrightarrow Q_u$$

für ein $u \in \mathbb{N}$ eine Ableitung gemäß G_1. Dann zeigen wir, daß für alle j, $j = 1, \ldots, u$, die Gleichung $Q_j = P_j P'_j$ mit $X_0 \Longrightarrow^*_G P_j$ und $X'_0 \Longrightarrow^*_{G'} P'_j$ gilt. Dies erfolgt durch Induktion. $Q_1 = X_0 X'_0$ erfüllt die Behauptung für $j = 1$. Es sei nun Q_j von der angegebenen Form. Da kein terminales Zeichen auf der linken Seite einer Produktion vorkommt und $V_N \cap V'_N = \emptyset$ gilt, hat auch Q_{j+1} diese Form. Somit können nur Wörter aus LL' erzeugt werden.

Es fehlt noch der Fall $i = 1$ mit $\varepsilon \in L \cup L'$. Wir setzen $L_1 = L - \{\varepsilon\}$, $L'_1 = L' - \{\varepsilon\}$. Für $L_1 L'_1$ kann nach den vorstehenden Überlegungen eine Typ-1-Grammatik konstruiert werden. LL' ist jetzt eine der folgenden Sprachen:

$$L_1 L'_1 \cup L'_1 \text{ (falls } \varepsilon \in L - L'),$$
$$L_1 L'_1 \cup L_1 \text{ (falls } \varepsilon \in L' - L),$$
$$L_1 L'_1 \cup L_1 \cup L'_1 \cup \{\varepsilon\} \text{ (falls } \varepsilon \in L \cap L').$$

Nach den vorher bewiesenen Aussagen ist LL' eine Typ-1-Sprache.

Schließlich betrachten wir den Abschluß unter Iteration. Für $i = 2$ konstruieren wir die Typ-2-Grammatik

$$G_1 = (V_N \cup \{Y_0\}, V_T, Y_0, F \cup \{Y_0 \to \varepsilon, Y_0 \to Y_0 X_0, Y_0 \to X_0\}).$$

Offenbar gilt $L(G_1) = L^*$. Wir bemerken, daß diese Konstruktion für die Typen 0 und 1 nicht richtig ist, da es in den Ableitungen illegitime Kontexte geben könnte. Eigentlich ist für $i = 2$ die Produktion $Y_0 \to X_0$ überflüssig, sie wird jedoch gebraucht, damit der Beweis von Satz 2.4 richtig wird.

Für $i = 3$ ersetzen wir in F jede Produktion

$$X \to w,\ X \in V_N, w \in V_T^*,$$

durch $X \to Y_0 w$ und erhalten damit eine Produktionenmenge F_2. Es ist

$$G_1 = (V_N \cup \{Y_0\}, V_T, Y_0, \{Y_0 \to \varepsilon, Y_0 \to X_0\} \cup F \cup F_2)$$

eine Typ-3-Grammatik mit $L(G_1) = L^*$. Eigentlich ist F in der Produktionenmenge von G_1 überflüssig, für den Beweis von Satz 2.4 ist F jedoch nötig.

Wir kommen zu den Fällen $i = 0$ und $i = 1$. Zu jeder Typ-0- oder Typ-1-Grammatik G_1 läßt sich eine ε-freie Grammatik G_1' vom selben Typ mit $L(G_1') = L(G_1) - \{\varepsilon\}$ konstruieren. Ist nämlich G_1 vom Typ 1, so wird einfach die Produktion $Y_0 \to \varepsilon$ entfernt, wobei Y_0 das Anfangssymbol ist. Ist G_1 vom Typ 0, so wird eine Produktion $P \to \varepsilon$ durch Produktionen

$$xP \to x \text{ und } Px \to x \text{ für alle } x \in V_N \cup V_T$$

ersetzt. Damit ist gesichert, daß ein Wort, das nur aus P besteht, nicht mehr gelöscht werden kann. Zu diesen Grammatiken können äquivalente ε-freie Grammatiken konstruiert werden, die die am Anfang des Beweises geforderten Eigenschaften erfüllen. Da für jede Sprache L die Gleichung $L^* = (L \cup \{\varepsilon\})^*$ gilt, reicht es zu zeigen, daß für jede Typ-i-Sprache L $(i = 0, 1)$ mit $\varepsilon \notin L$ auch die Sprache L^* vom Typ i ist.

Ohne Beschränkung der Allgemeinheit gelte also $\varepsilon \notin L$. Wir konstruieren die Typ-i-Grammatik

$$\begin{aligned}\bar{G} = (V_N \cup \{Y_0, Y_1\}, V_T, Y_0, F \cup \{Y_0 \to \varepsilon, Y_0 \to X_0, Y_0 \to Y_1 X_0\} \\ \cup \{Y_1 a \to Y_1 X_0 a \mid a \in V_T\} \cup \{Y_1 a \to X_0 a \mid a \in V_T\}).\end{aligned}$$

Es gilt $L(\bar{G}) = L^*$. Dies sehen wir wie folgt ein. Zunächst wird ersichtlich jedes Wort aus L^* durch $\bar{G}$ erzeugt. Andererseits betrachten wir eine Ableitung

$$Y_0 \Longrightarrow_{\bar{G}} Q_1 \Longrightarrow_{\bar{G}} \ldots \Longrightarrow_{\bar{G}} Q_u \text{ für } u \in \mathbb{N}.$$

Ist nun $Q_1 = \varepsilon$ oder $Q_1 = X_0$, so ist $Q_1 = \varepsilon \in L^*$, oder die Ableitung kann von X_0 aus nur zu terminalen Wörtern aus L führen. Als weitere Möglichkeit kommt nur noch $Q_1 = Y_1 X_0$ in Betracht, wobei Q_j, $j = 1, \ldots, u$, von einer der beiden folgenden Formen ist:

(1) $Q_j = Y_1 R_1 \ldots R_v$, $v \in \mathbb{N}$, wobei das erste Symbol in den Wörtern $R_2, \ldots, R_v$ terminal ist und $X_0 \Longrightarrow_G^* R_\nu$ für alle $\nu = 1, \ldots, v$ gilt,

(2) $Q_j = R_0 R_1 \ldots R_v$, $v \in \mathbb{N}$, wobei das erste Symbol in den Wörtern $R_1, \ldots, R_v$ terminal ist und $X_0 \Longrightarrow_G^* R_\nu$ für alle $\nu = 0, 1, \ldots, v$ gilt.

Eventuelle Nichtterminalzeichen in R_ν und $R_{\nu+1}$ werden durch das erste Zeichen von $R_{\nu+1}$, das jeweils ein Terminalzeichen ist, getrennt. Die Aussage über Q_j beweisen wir durch Induktion. Offenbar ist Q_1 von der Form (1). Nach Induktionsannahme sei nun Q_j ein Wort der Form (1) oder (2). Da auf der linken Seite einer Produktion aus F keine Terminalzeichen vorkommen, so wird aus einem Wort der Form (1) bei Anwendung von Produktionen aus F, aber auch bei Anwendung von Produktionen der vorletzten Produktionenmenge von $\bar{G}$, ein Wort Q_{j+1} der Form (1) erzeugt. Bei Anwendung einer Produktion der letzten Produktionenmenge geht es jedoch in ein Wort der Form (2) über. Ist Q_j ein Wort der Form (2), so wird daraus bei Anwendung irgendeiner Produktion ein Wort Q_{j+1} der Form (2). Wir sehen also, daß Q_{j+1} ein Wort der Form (1) oder (2) ist. Damit ist der Induktionsbeweis beendet. Es folgt, daß mit $\bar{G}$ nur Wörter aus L^* ableitbar sind. □

Satz 2.3 liefert in allen Fällen einen Algorithmus zur Konstruktion der jeweiligen Grammatik. In fast allen folgenden Beweisen wird ein Algorithmus geliefert.

Eine unmittelbare Folgerung aus dem Beweis von Satz 2.3 ist

Satz 2.4 Es sei $L \in \mathfrak{L}_i$ für $i = 0, 1, 2$ oder 3. Dann gilt $L^+ \in \mathfrak{L}_i$.

Beweis: Es muß nur die Produktion $Y_0 \to \varepsilon$ aus den L^* erzeugenden Grammatiken im Beweis von Satz 2.3 entfernt werden. □

Satz 2.5 $\mathfrak{L}_0$ und $\mathfrak{L}_2$ sind abgeschlossen unter Substitution und folglich auch unter beliebigem Homomorphismus.

Beweis: Es sei $L = L(G)$ mit $G = (V_N, V_T, X_0, F)$ eine Grammatik vom Typ j, $j = 0$ oder $j = 2$. Dabei gelte $V_T = \{a_1, \ldots, a_r\}$ für ein $r \in \mathbb{N}$. Weiter sei σ eine Substitution, wobei $\sigma(a_i)$ von einer Typ-j-Grammatik ($j = 0, 2$) $G_i = (V_N^i, V_T^i, X_0^i, F^i)$, $i = 1, \ldots, r$, erzeugt wird. Ohne Beschränkung der Allgemeinheit gelte $V_N^i \cap V_N^{i'} = \emptyset$ für $i \neq i'$ und $V_N^i \cap V_N = \emptyset$. Außerdem komme nach Satz 2.2 auf der linken Seite einer Produktion kein terminales Zeichen vor.

Für $j = 2$ ersetzen wir die Symbole a_i, $i = 1, \ldots, r$, in den Produktionen von F durch X_0^i und erhalten so eine Produktionenmenge F_1. Wir definieren damit eine Typ-2-Grammatik durch

$$\bar{G} = (V_N \cup V_N^1 \cup \ldots \cup V_N^r, V_T^1 \cup \ldots \cup V_T^r, X_0, F_1 \cup F^1 \cup \ldots \cup F^r).$$

Es gilt offenbar $L(\bar{G}) = \sigma(L)$.

Diese Definition ist für $j = 0$ wegen des möglichen Entstehens illegitimer Kontexte (nur möglich, falls a_i^2 ein Teilwort eines Wortes von L ist) im allgemeinen nicht richtig. Wir konstruieren eine Typ-0-Grammatik $G_\sigma = (V_N^\sigma, V_T^1 \cup \ldots \cup V_T^r, Y_0, F^\sigma)$ mit $L(G_\sigma) = \sigma(L)$ wie folgt. Es sei

$$V_N^\sigma = V_N \cup V_N^1 \cup \ldots \cup V_N^r \cup \{Y_0, Y, A_{a_1}, \ldots, A_{a_r}\}.$$

Zur Konstruktion von F^σ ersetzen wir zunächst in F jedes Symbol a_i, $i = 1, \ldots, r$, durch A_{a_i}. Die so entstandene Produktionenmenge nennen wir F_2. Weiter sei $F_3 =$

$\{YA_{a_i} \to YX_0^i \mid i = 1, \ldots, r\}$, $F_4 = \{Ya \to aY \mid a \in V_T^1 \cup \ldots \cup V_T^r\}$. Damit definieren wir

$$F^\sigma = F_2 \cup F_3 \cup F_4 \cup F^1 \cup \ldots \cup F^r \cup \{Y_0 \to YX_0, Y \to \varepsilon\}.$$

Wir betrachten $w \in \sigma(L)$. Dann existiert ein Wort $v = a_{i_1} \ldots a_{i_m} \in L$, $i_\mu \in \{1, \ldots, r\}$, $\mu = 1, \ldots, m$, $m \in \mathbb{N}_0$, mit $w = w_{i_1} \ldots w_{i_m}$, $w_{i_\mu} \in \sigma(a_{i_\mu})$. Wir geben eine Ableitung von w gemäß G_σ an. Ist $v = \varepsilon$, so erhalten wir die Ableitung

$$Y_0 \Longrightarrow YX_0 \Longrightarrow_{F_2}^* Y \Longrightarrow \varepsilon.$$

Anderenfalls gilt

$$\begin{aligned} Y_0 &\Longrightarrow YX_0 \Longrightarrow_{F_2}^* YA_{a_{i_1}} \ldots A_{a_{i_m}} \Longrightarrow_{F_3} YX_0^{i_1} A_{a_{i_2}} \ldots A_{a_{i_m}} \Longrightarrow_{F^{i_1}}^* Yw_{i_1} A_{a_{i_2}} \ldots A_{a_{i_m}} \\ &\Longrightarrow_{F_4}^* w_{i_1} YA_{a_{i_2}} \ldots A_{a_{i_m}} \Longrightarrow_{F_3} \ldots \Longrightarrow_{F^{i_m}}^* w_{i_1} \ldots w_{i_{m-1}} Yw_{i_m} \Longrightarrow w. \end{aligned}$$

Umgekehrt werden nur Wörter aus $\sigma(L)$ gemäß G_σ erzeugt. Dabei können die oben angegebenen Ableitungsschritte zum Teil vermischt werden. Da jedoch die Symbole A_{a_i} außer bei F_3 auf den linken Seiten der Produktionen nicht auftreten und es keine Terminalzeichen auf den linken Seiten der Produktionen aus F^i gibt, entstehen dabei keine illegitimen Kontexte. Das von links nach rechts wandernde Y stößt jeweils durch Produktionen aus F_3 die Ableitung der Grammatiken G_{i_μ} $(\mu = 1, \ldots, m)$ an, wobei das Wandern von Y durch das Vertauschen gemäß F_4 erst möglich ist, wenn G_{i_μ} hinter dem Y mindestens ein Terminalsymbol erzeugt hat. Erzeugt G_{i_μ} das leere Wort, so entfällt das Vertauschen. □

Der Satz gilt auch für $\mathfrak{L}_3$ (siehe Satz 4.2) und im Falle der ε-freien Substitution auch für $\mathfrak{L}_1$ (siehe Satz 9.5).

Satz 2.6 $\mathfrak{L}_0$ ist abgeschlossen unter Durchschnittsbildung.

Beweis: Es seien $G = (V_N, V_T, X_0, F)$ und $G' = (V_N', V_T', X_0', F')$ Typ-0-Grammatiken mit $L = L(G)$, $L' = L(G')$. Ohne Beschränkung der Allgemeinheit gelte $V_N \cap V_N' = \emptyset$.

Wir setzen

$$G_1 = (V_N \cup V_N' \cup V_N'', V_T \cup V_T', Y_0, F \cup F' \cup F_1)$$

mit $V_N'' = \{A_a \mid a \in V_T \cup V_T'\} \cup \{Y_0, Y_1, Y_2\}$. F_1 besteht aus folgenden Produktionen:

(1) $Y_0 \to Y_1 X_0 Y_2 X_0' Y_1$,
(2) $Y_2 a \to A_a Y_2$ für $a \in V_T \cup V_T'$,
(3) $bA_a \to A_a b$ für $a, b \in V_T \cup V_T'$,
(4) $Y_1 A_a a \to a Y_1$ für $a \in V_T \cup V_T'$,
(5) $Y_1 Y_2 Y_1 \to \varepsilon$.

Mit (1) und Produktionen aus $F \cup F'$ folgt

$$Y_0 \Longrightarrow Y_1 X_0 Y_2 X_0' Y_1 \Longrightarrow^* Y_1 w Y_2 w' Y_1$$

für beliebige $w \in L$, $w' \in L'$. Wegen (2) bis (4) ist daraus weiter $wY_1Y_2Y_1$ genau dann ableitbar, wenn $w = w'$ gilt. Diese Ableitungsschritte gemäß (2) bis (4) können zum Teil schon erfolgen, bevor $Y_1wY_2w'Y_1$ erzeugt ist. Wegen unserer Annahmen gibt es dabei keine illegitimen Kontexte. Mit (5) wird schließlich w aus $wY_1Y_2Y_1$ abgeleitet. Dies ist der einzige Fall, daß eine Ableitung von Y_0 aus zu einem terminalen Wort führt. Folglich gilt $L(G_1) = L \cap L'$. $\square$

Wir werden später sehen, daß ein entsprechendes Ergebnis zwar für $\mathfrak{L}_3$ (siehe Satz 3.3 mit Satz 1.2) und $\mathfrak{L}_1$ (siehe Satz 9.5) gilt, nicht jedoch für $\mathfrak{L}_2$ (siehe Satz 5.12).

2.3 Monotone und Typ-1-Grammatiken

Wir beweisen jetzt, daß monotone und Typ-1-Grammatiken äquivalent sind.

Satz 2.7 Zu jeder monotonen Grammatik existiert eine äquivalente Typ-1-Grammatik.

Beweis: Zunächst zeigen wir, wie eine monotone Produktion durch mehrere Typ-1-Produktionen ersetzt werden kann.

Es sei $G_1 = (V_N, V_T, X_0, F)$ eine Typ-1-Grammatik mit $(X_0, \varepsilon) \notin F$. Weiter gelte $P, Q \in V_N^*$ mit

$$\begin{aligned} &P = X_1 \dots X_m,\ Q = Y_1 \dots Y_n,\ X_i, Y_j \in V_N, \\ &\quad i = 1, \dots, m,\ j = 1, \dots, n,\ m, n \in \mathbb{N},\ 2 \le m \le n. \end{aligned}$$

Wir setzen $G_2 = (V_N, V_T, X_0, F \cup \{P \to Q\})$. Wir werden sehen, daß G_2 zur Typ-1-Grammatik

$$G_2' = (V_N \cup \{Z_1, \dots, Z_m\}, V_T, X_0, F \cup F')$$

äquivalent ist, wobei F' aus den Produktionen

$$\begin{aligned} X_1X_2 \dots X_m &\to Z_1X_2 \dots X_m, \\ Z_1X_2 \dots X_m &\to Z_1Z_2X_3 \dots X_m, \\ &\vdots \\ Z_1Z_2 \dots Z_{m-1}X_m &\to Z_1Z_2 \dots Z_{m-1}Z_mY_{m+1} \dots Y_n, \\ Z_1Z_2 \dots Z_mY_{m+1} \dots Y_n &\to Y_1Z_2 \dots Z_mY_{m+1} \dots Y_n, \\ &\vdots \\ Y_1 \dots Y_{m-1}Z_mY_{m+1} \dots Y_n &\to Y_1 \dots Y_mY_{m+1} \dots Y_n \end{aligned}$$

besteht. Man beachte, daß bei allen Produktionen bis auf $Z_1Z_2 \dots Z_{m-1}X_m \to Z_1Z_2 \dots Z_{m-1}Z_mY_{m+1} \dots Y_n$ die linke und die rechte Seite dieselbe Länge haben. G_2' ist offensichtlich vom Typ 1, und es gilt $L(G_2) \subset L(G_2')$. Umgekehrt können die Symbole Z_i nur durch die erste der angegebenen Produktionen von F' eingeführt werden, und die ganze Folge dieser Produktionen muß genau in dieser Reihenfolge angewendet werden, um die Nichtterminalzeichen Z_i schließlich zu entfernen. Zwischendurch können

eventuell schon Produktionen aus F mit Symbolen X_i bzw. mit Y_j auf der linken Seite angewendet werden. Nur wenn nach einer endlichen Zahl von Ableitungsschritten dabei der richtige Kontext wiederhergestellt ist, kann die Folge aus F' fortgesetzt werden. Diese Ersetzungen mit Produktionen aus F, die mit einer Produktion mit X_i bzw. Y_j auf der linken Seite beginnen, können aber auch vor bzw. nach der Simulierung der Produktion $P \to Q$ durchgeführt werden, ohne die erzeugte Sprache zu verändern. Es gilt also insgesamt $L(G_2') = L(G_2)$.

Es sei jetzt $G = (V_N, V_T, X_0, F)$ eine beliebige monotone Grammatik. Ohne Beschränkung der Allgemeinheit nehmen wir nach Satz 2.2 an, daß Terminalzeichen höchstens in Produktionen $X \to a$ mit $X \in V_N$ und $a \in V_T$ vorkommen. Zunächst entfernen wir die eventuell vorhandene Produktion $X_0 \to \varepsilon$ und alle Produktionen, die nicht vom Typ 1 sind, aus F. Die so bestimmte Grammatik nennen wir G_1. Zu dieser fügen wir eine der zuvor entfernten Produktionen hinzu und erhalten eine Grammatik G_2. Mit G_2 führen wir die obige Konstruktion durch, und es ergibt sich eine äquivalente Typ-1-Grammatik G_2'. Durch Hinzufügen einer weiteren zuvor entfernten Produktion zu G_2' wird das Verfahren fortgesetzt, bis schließlich alle Produktionen von G berücksichtigt sind. Es ergibt sich eine Typ-1-Grammatik G' mit $L(G') = L(G) - \{\varepsilon\}$. Falls $X_0 \to \varepsilon$ in F liegt, erhalten wir mit einem neuen Anfangssymbol X_0' und den zusätzlichen Produktionen $X_0' \to \varepsilon$ und $X_0' \to X_0$ eine Typ-1-Grammatik G'' mit $L(G'') = L(G)$. $\square$

Kapitel 3

Automaten und Sprachen

Nachdem in Kapitel 1 endliche erkennende Automaten und in Kapitel 2 Sprachen und Grammatiken behandelt wurden, soll in diesem Kapitel der Zusammenhang zwischen den verschiedenen Familien von Sprachen und den endlichen erkennenden Automaten bzw. den noch zu definierenden Klassen von Automaten aufgezeigt werden.

3.1 Endliche erkennende Automaten und Typ-3-Sprachen

Satz 3.1 Es sei $G = (V_N, V_T, X_0, F)$ eine analysierende Grammatik vom Typ 3. Dann existiert eine äquivalente analysierende Typ-3-Grammatik $G' = (V'_N, V_T, X_0, F')$, deren Produktionen eine der folgenden Formen haben:

(a) (Xa, Y) mit $X, Y \in V'_N$, $a \in V_T$,
(b) (ε, Y) mit $Y \in V'_N$.

Beweis: Die Grammatik G kann außer Produktionen der Form (a) oder (b) auch noch Produktionen der folgenden Formen haben:

(c) $(Ya_1 \dots a_k, X)$ mit $k \in \mathbb{N}$, $k \geq 2$, $X, Y \in V_N$, $a_j \in V_T$,
(d) $(a_1 \dots a_k, Y)$ mit $k \in \mathbb{N}$, $k \geq 1$, $Y \in V_N$, $a_j \in V_T$,
(e) (Y, X') mit $X', Y \in V_N$.

Für jede Produktion (c) führen wir neue Nichtterminalzeichen $X_1, \dots, X_{k-1}$ ein und ersetzen (c) durch

$$(Ya_1, X_1), (X_1a_2, X_2), \dots, (X_{k-1}a_k, X).$$

Für jede Produktion (d) werden neue Nichtterminalzeichen $Y_1, \dots, Y_k$ eingeführt und (d) durch

$$(\varepsilon, Y_1), (Y_1a_1, Y_2), \dots, (Y_ka_k, Y)$$

ersetzt. Damit erhalten wir offenbar eine analysierende Typ-3-Grammatik G'' mit $L(G'') = L(G)$.

Es müssen noch die Produktionen der Form (e) ersetzt werden. Wir definieren

$$U(Z) = \{Z' \mid Z \Longrightarrow^* Z',\ Z' \in V_N''\} \text{ für alle } Z \in V_N''.$$

Es gilt $Z \in U(Z)$. $U(Z)$ kann effektiv bestimmt werden, da es nur endlich viele Nichtterminalzeichen in G'' gibt. Die Produktionen der Form (e) werden dadurch berücksichtigt, daß wir für jede Produktion (Xa, Y) der Form (a) noch die Produktionen

$$(Xa, Y') \text{ für alle } Y' \in U(Y)$$

und für jede Produktion (ε, Y) der Form (b)

$$(\varepsilon, Y') \text{ für alle } Y' \in U(Y)$$

aufnehmen. Die so bestimmte Grammatik G' erfüllt die Behauptung des Satzes. □

Aufgrund der Dualität folgt

Satz 3.2 Es sei $G = (V_N, V_T, X_0, F)$ eine Grammatik vom Typ 3. Dann existiert eine äquivalente Typ-3-Grammatik $G' = (V_N', V_T, X_0, F')$, deren Produktionen eine der folgenden Formen haben:
(a) (X, Ya) mit $X, Y \in V_N'$, $a \in V_T$,
(b) (Y, ε) mit $Y \in V_N'$. □

Der folgende Satz ist zum Teil schon in Satz 1.1 enthalten. Wir geben hier einen zweiten Beweis für diesen Satz.

Satz 3.3 Es sei L eine Sprache. Die folgenden Aussagen sind äquivalent:
(a) L ist von einem endlichen erkennenden Automaten akzeptierbar.
(b) L ist von einem nichtdeterministischen endlichen erkennenden Automaten akzeptierbar.
(c) L ist vom Typ 3.

Beweis: Aus (a) folgt (b), da ein deterministischer endlicher erkennender Automat ein Spezialfall eines nichtdeterministischen Automaten ist.

Wir beweisen nun, daß aus (b) die Aussage (c) folgt. Es sei $N = (S \cup V_T, F)$ ein nichtdeterministischer endlicher erkennender Automat. Dann gilt

$$L(N) = \bigcup_{\substack{s_0 \in S_0 \\ s_1 \in S_1}} \{w \mid w \in V_T^*,\ s_0 w \Longrightarrow^* s_1\}.$$

Die durch ein festes Paar (s_0, s_1) bestimmte Menge dieser Vereinigung wird durch die analysierende Typ-3-Grammatik $G_{s_0,s_1} = (S, V_T, s_1, F \cup \{(\varepsilon, s_0)\})$ erkannt. Dabei ist zu beachten, daß eine Ableitung $s_0 w \Longrightarrow^* s_1$ gemäß N auch eine Ableitung gemäß G_{s_0,s_1} ist. Bei der analysierenden Grammatik geht dieser Ableitung noch der Ableitungsschritt $w \Longrightarrow s_0 w$ mit Hilfe der Produktion (ε, s_0) voraus. Man beachte, daß dabei

das leere Wort an der linken Seite von w ($w = \varepsilon w$) durch s_0 ersetzt wird, anderenfalls würde man am Ende keinen Zustand erhalten. Nach Satz 2.3 ist die Vereinigung zweier Typ-3-Grammatiken wieder vom Typ 3. Folglich ist $L(N)$ eine Typ-3-Sprache.

Schließlich zeigen wir, daß aus (c) die Aussage (a) folgt. Nach Satz 3.1 können wir ohne Beschränkung der Allgemeinheit annehmen, daß $G = (V_N, V_T, X_0, F)$ eine analysierende Grammatik ist mit $L = L(G)$ und Produktionen der Form (Xa, Y) oder (ε, X) für $X, Y \in V_N$, $a \in V_T$. Zu konstruieren ist ein endlicher erkennender Automat $E = (S \cup V_T, F_1)$. Wir setzen $S = \mathfrak{P}(V_N)$ mit Anfangszustand $Z_0 = \{Y \mid (\varepsilon, Y) \in F\} \subset V_N$ und Endzustandsmenge $S_1 = \{Z \mid Z \subset V_N, X_0 \in Z\}$. Die Produktionenmenge wird durch

$$F_1 = \{(Za, Z_1) \mid Z \in S, a \in V_T, Z_1 = \{Y \mid (Xa, Y) \in F, X \in Z\}\}$$

definiert. Zu zeigen ist $L(G) = L(E)$.

Für $w = \varepsilon$ gilt die Äquivalenz

$$\varepsilon \in L(G) \iff (\varepsilon, X_0) \in F \iff X_0 \in Z_0 \iff Z_0 \in S_1 \iff \varepsilon \in L(E).$$

Für $w = a_1 \dots a_n$, $a_i \in V_T$, $i = 1, \dots, n$, $n \in \mathbb{N}$, folgt aus $w \in L(G)$ die Existenz einer Ableitung

$$(*)\ a_1 \dots a_n \Longrightarrow_G Y_1 a_1 \dots a_n \Longrightarrow_G Y_2 a_2 \dots a_n \Longrightarrow_G^* Y_{n-1} a_{n-1} a_n \Longrightarrow_G Y_n a_n \Longrightarrow_G X_0$$

in G mit Produktionen

$$\varepsilon \to Y_1, Y_1 a_1 \to Y_2, \dots, Y_{n-1} a_{n-1} \to Y_n, Y_n a_n \to X_0$$

und geeigneten Zeichen $Y_1, \dots, Y_n \in V_N$. Im Fall $n = 1$ wird $X_0 = Y_2$ gesetzt. Wegen $(\varepsilon, Y_1) \in F$ und der Definition von E gilt $Y_1 \in Z_0$. Mit dem Wort w und dem Zustand Z_0 des Automaten E betrachten wir dann die eindeutig bestimmte Ableitung

$$(**)\ Z_0 a_1 a_2 \dots a_n \Longrightarrow_E Z_1 a_2 \dots a_n \Longrightarrow_E^* Z_{n-2} a_{n-1} a_n \Longrightarrow_E Z_{n-1} a_n \Longrightarrow_E Z_n.$$

Wegen $Y_1 \in Z_0$, $(Y_1 a_1, Y_2) \in F$ und der Definition von F_1 erhalten wir $Y_2 \in Z_1$. Falls $n \geq 2$ ist, ergibt sich wegen $(Y_2 a_2, Y_3) \in F$ weiter $Y_3 \in Z_2$ usw., bis schließlich $X_0 \in Z_n$ folgt wegen $Y_n \in Z_{n-1}$, $(Y_n a_n, X_0) \in F$. Die Definition von S_1 liefert $Z_n \in S_1$ und damit $w \in L(E)$.

Für $w \in L(E)$ gibt es umgekehrt eine Ableitung $(**)$ mit Produktionen $(Z_0 a_1, Z_1), \dots, (Z_{n-1} a_n, Z_n) \in F_1$ und $Z_n \in S_1$. Dann ist $X_0 \in Z_n$. Nach Definition von F_1 existiert $Y_n \in Z_{n-1}$ mit $Y_n a_n \to X_0$ gemäß G. Falls $n \geq 2$ ist, gibt es zu Y_n ein Symbol $Y_{n-1} \in Z_{n-2}$ mit einer Produktion $Y_{n-1} a_{n-1} \to Y_n$ usw. In jedem Fall erhalten wir am Ende $Y_1 \in Z_0$ mit einer Produktion $Y_1 a_1 \to Y_2$. Aus $Y_1 \in Z_0$ folgt, daß auch $\varepsilon \to Y_1$ eine Produktion in G ist. Somit erhalten wir eine Ableitung $(*)$, und folglich ist $w \in L(G)$. Insgesamt haben wir damit $L(E) = L(G)$ bewiesen. □

Zum Beweis der Äquivalenz von (b) und (c) wäre es einfacher gewesen zu zeigen, daß aus (c) die Aussage (b) folgt. Oben haben wir jedoch gleichzeitig einen weiteren Beweis für Satz 1.1 geliefert. Der Beweis von Satz 1.1 wurde in Kapitel 1 nicht vollständig durchgeführt.

3.2 Endliche Übersetzungsautomaten

Bisher haben wir nur solche Automaten betrachtet, die „ja“ oder „nein“ als Ausgabe haben, und zwar je nachdem, ob sie sich nach Abarbeitung des Wortes in einem Endzustand befinden oder nicht. Im folgenden sollen endliche Automaten betrachtet werden, die bei Eingabe eines Symbols ein Wort über dem Ausgabealphabet ausgeben. In der englischsprachigen Literatur werden solche Automaten auch „generalized sequential machines“ genannt.

Definition 3.1 Es sei $U = (V, F)$ ein kombinatorisches System. U heißt *endlicher Übersetzungsautomat*, wenn die folgenden Eigenschaften erfüllt sind.

(a) Es gilt $V = S \cup V_I \cup V_O$ mit *Zustandsalphabet* S, *Eingabealphabet* V_I und *Ausgabealphabet* V_O. Dabei ist $S \cap (V_I \cup V_O) = \emptyset$.

(b) Es wird ein *Anfangszustand* $s_0 \in S$ sowie eine *Menge von Endzuständen* $S_1 \subset S$ ausgezeichnet.

(c) Die *Produktionen* von F sind von der Form $s_i a \to v s_j$ für $s_i, s_j \in S$, $a \in V_I$ und $v \in V_O^*$. □

Definition 3.2 Es sei $U = (V, F)$ ein endlicher Übersetzungsautomat.

(a) Es sei $w \in V_I^*$. Dann heißt

$$U(w) = \{v \mid s_0 w \Longrightarrow^* v s_1,\ s_1 \in S_1\}$$

die *Menge der Übersetzungen* von w.

(b) Es sei $L \subset V_I^*$. U *übersetzt* L (*bildet* L *ab*) in

$$U(L) = \{v \mid v \in U(w), w \in L\}.$$

Die so definierte Abbildung $U : \mathfrak{P}(V_I^*) \to \mathfrak{P}(V_O^*)$ heißt *EÜA-Abbildung*.

(c) Es sei $v \in V_O^*$. Dann heißt

$$U^{-1}(v) = \{w \mid v \in U(w)\}$$

die *inverse Menge der Übersetzungen* von v.

(d) Es sei $L \subset V_O^*$. U *übersetzt* L *invers* (*bildet* L *invers ab*) in

$$U^{-1}(L) = \{w \mid w \in U^{-1}(v), v \in L\}.$$

Die so definierte Abbildung von $U^{-1} : \mathfrak{P}(V_O^*) \to \mathfrak{P}(V_I^*)$ heißt *inverse EÜA-Abbildung*. □

Man beachte, daß ein endlicher Übersetzungsautomat U und die von ihm definierte EÜA-Abbildung mit demselben Symbol U bezeichnet werden.

Definition 3.3 Es sei U ein endlicher Übersetzungsautomat. U heißt *deterministischer endlicher Übersetzungsautomat*, wenn für alle $(s_i, a) \in S \times V_I$ genau eine Produktion $(s_i a, v s_j) \in F$ existiert. □

Definition 3.4 Es sei U ein deterministischer endlicher Übersetzungsautomat. U heißt *Mealy-Automat*, wenn für alle Produktionen $(s_i a, v s_j) \in F$ die Gleichung $|v| = 1$ gilt. □

Wenn U_M ein Mealy-Automat ist, dann ist die nach Definition 3.2(a) durch U_M gegebene partielle Abbildung längenerhaltend, es gilt also

$$|U_M(w)| = |w| \text{ für alle } w \in V_I^*, \text{ für die } U_M(w) \text{ definiert ist.}$$

Ist U ein deterministischer endlicher Übersetzungsautomat, dann bestimmen die Produktionen $(s_i a, v s_j)$ zwei Abbildungen

$$\delta : S \times V_I \to S \text{ und } \lambda : S \times V_I \to V_O^* \text{ mit } \delta(s_i, a) = s_j \text{ und } \lambda(s_i, a) = v,$$

die *Überführungsfunktion* bzw. *Ausgabefunktion*. Bei Mealy-Automaten erhalten wir speziell eine Ausgabefunktion $\lambda : S \times V_I \to V_O$.

Eine entsprechende Bezeichnung mit einer Überführungsfunktion $\delta : S \times V_T \to S$ ist auch für endliche erkennende Automaten üblich. Man schreibt dann $(S, V_T, \delta, s_0, S_1)$, wie wir es auch schon auf Seite 6 erwähnt haben

Es existieren (siehe [24], S. 31/32) ein endlicher Übersetzungsautomat U sowie Sprachen L und L' mit

$$U^{-1}(U(L)) \neq L, \; U(U^{-1}(L')) \neq L'.$$

Dies zeigt, daß U^{-1} keine echte Inverse im algebraischen Sinn ist.

In den vorstehenden Überlegungen wurden endliche Übersetzungsautomaten als Sprachübersetzer eingeführt. Sie können aber auch als ein Erkennungsverfahren aufgefaßt werden. Für einen Automaten U werde die von U akzeptierte Sprache durch

$$L(U) = \{w \in V_I^* \mid U(w) \neq \emptyset\}$$

gegeben. Da das genaue Übersetzungsergebnis ignoriert wird, ist $L(U)$ bereits durch den zugrundeliegenden nichtdeterministischen endlichen erkennenden Automaten festgelegt, den man dadurch erhält, daß man eine Produktion $(s_i a, v s_j)$ durch $(s_i a, s_j)$ ersetzt. Das bedeutet, daß $L(U)$ eine Typ-3-Sprache ist, und umgekehrt kann jede Typ-3-Sprache auch durch einen endlichen Übersetzungsautomaten akzeptiert werden.

3.3 Kellerautomaten und Typ-2-Sprachen

Die Möglichkeit der Familie der endlichen erkennenden Automaten, Sprachen zu akzeptieren, kann erweitert werden, indem man das „Gedächtnis“ vergrößert, also die Endlichkeit der Zustandsmenge aufgibt. Um aber Automaten mit „unendlich vielen Zuständen“ noch überschaubar bleiben zu lassen, muß das Hinzunehmen von Zuständen geeignet endlich beschrieben werden. Dies läßt sich durch das Anfügen von potentiell unendlich großen Speicherbändern erreichen, die aber für die verschiedenen zu definierenden Typen verschiedene Zugriffsarten aufweisen. In diesem Abschnitt werden wir Kellerautomaten betrachten.

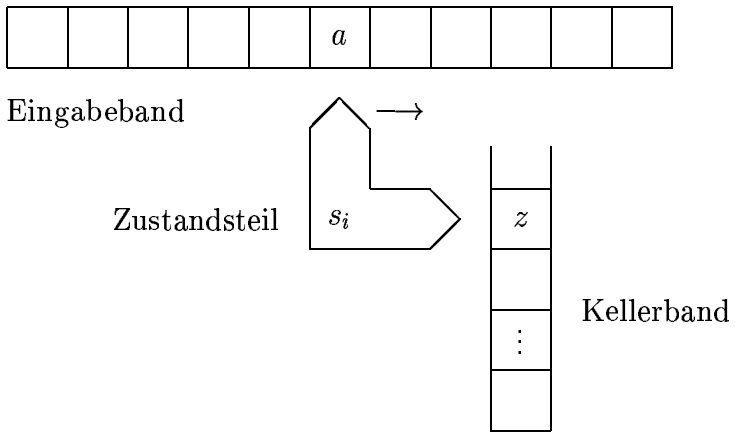
Definition 3.5 Es sei $K = (V, F)$ ein kombinatorisches System. K heißt *Kellerautomat*, wenn die folgenden Aussagen erfüllt sind:

(a) Es gilt $V = S \cup V_I \cup V_Z$ mit *Zustandsalphabet* S, *Eingabealphabet* V_I und *Kelleralphabet* V_Z, wobei $S \cap (V_I \cup V_Z) = \emptyset$ ist.

(b) Es werden ein *Anfangszustand* $s_0 \in S$, ein *Startzeichen* $z_0 \in V_Z$ des Kellerbandes und eine *Menge von Endzuständen* $S_1 \subset S$ ausgezeichnet.

(c) Die Produktionen von F sind von einer der beiden Formen:
   (α) $zs_i \to Qs_j$ mit $z \in V_Z$, $Q \in V_Z^*$, $s_i, s_j \in S$,
   (β) $zs_ia \to Qs_j$ mit $z \in V_Z$, $Q \in V_Z^*$, $s_i, s_j \in S$, $a \in V_I$. □

Einen Kellerautomaten können wir uns durch das folgende Bild veranschaulichen:



Ein Kellerautomat besteht aus einem Eingabeband, einem Kellerband, einem Zustandsteil, einem Lesekopf für das Eingabeband sowie einem Lese- und Schreibkopf für das Kellerband. Das Eingabeband enthält das jeweilige Eingabewort w, das von links nach rechts gelesen wird. Befindet sich der Kellerautomat im Zustand $s_i \in S$, ist $z \in V_Z$ das oberste Kellerelement und steht unter seinem Lesekopf das Symbol $a \in V_I$, so kann er entweder a lesen oder auch ignorieren. Im ersten Fall geht der Automat nichtdeterministisch gemäß der Regel (β) in einen Zustand s_j über und ersetzt das oberste Kellerelement z durch Q, wobei auch $Q = \varepsilon$ gelten darf. Ist $Q = z_1 \dots z_r$, $r \in \mathbb{N}$, dann wird z_r das oberste Kellerelement. Der Kopf über dem Eingabeband geht anschließend zum nächsten Eingabezeichen weiter. Das Symbol a dieser Stelle ist künftig nicht mehr erreichbar. Im zweiten Fall wählt der Kellerautomat eine Instruktion (α). Die Zustandsänderung und das Beschreiben des Kellerbandes erfolgt wie im ersten Fall, jedoch bleibt der Kopf an seinem Platz. Wir sehen, daß der Kellerautomat nur auf das jeweils oberste Element des Kellerbandes zugreifen kann.

Definition 3.6 (a) Es sei $K = (V, F)$ ein Kellerautomat. Mit

$$L(K) = \{w \mid w \in V_I^*,\ z_0 s_0 w \Longrightarrow^* Qs_1,\ Q \in V_Z^*, s_1 \in S_1\}$$

wird die von K *angenommene* oder *erkannte (akzeptierte)* Sprache bezeichnet.

(b) Es sei $L \subset V_I^*$ eine Sprache. L heißt *von einem Kellerautomat erkennbar (akzeptierbar)*, wenn ein Kellerautomat K existiert mit $L(K) = L$. □

Ein Wort w wird von einem Kellerautomaten akzeptiert, wenn der Automat, mit dem Anfangszustand s_0 und dem Startzeichen z_0 beginnend, nach Abarbeitung von w in einen Endzustand übergegangen ist. Dabei beginnt die Bearbeitung von w mit dem ersten Symbol von w. Sie ist beendet, wenn der Lesekopf rechts von w steht, also das letzte Symbol von w abgearbeitet wurde.

Ohne Beweis (siehe z.B. [3], Satz 2.3.15, [21], S. 70–80) zitieren wir den folgenden wichtigen Satz:

Satz 3.4 Eine Sprache wird genau dann von einem Kellerautomaten erkannt, wenn sie vom Typ 2 ist. □

Wir bemerken, daß wir diesen Satz in den folgenden Beweisen nicht benötigen.

Die durch Definition 3.3 gegebenen Kellerautomaten sind offenbar nichtdeterministisch. Wir definieren jetzt deterministische Automaten.

Definition 3.7 Es sei $K = (V, F)$ ein Kellerautomat. K heißt *deterministischer Kellerautomat*, wenn die folgenden Aussagen gelten:

(a) Für jedes $(s_i, z) \in S \times V_Z$ enthält F genau eine Produktion (α) und keine Produktion (β) oder aber keine Produktion (α) und für alle $a \in V_I$ genau eine Produktion (β).

(b) Gilt $z = z_0$ in (α) oder (β), so ist z_0 das erste Symbol von Q. □

Die Forderung (a) ist die eigentliche Forderung der Determiniertheit, die Forderung (b) sichert, daß der Keller niemals leer wird. Auf (b) könnte auch verzichtet werden.

Definition 3.8 Eine Sprache L heißt *deterministische Typ-2-Sprache*, wenn ein deterministischer Kellerautomat K mit $L(K) = L$ existiert.

$$\mathfrak{L}_{2,d} = \{L \mid L \text{ deterministische Typ-2-Sprache}\}$$

heißt *Familie der deterministischen Typ-2-Sprachen.* □

Satz 3.5 Es existieren Typ-2-Sprachen, die keine deterministischen Typ-2-Sprachen sind. □

Unter Annahme der Gültigkeit von Satz 3.4 folgt aus diesem Satz die echte Inklusion $\mathfrak{L}_{2,d} \subsetneq \mathfrak{L}_2$. Der Beweis von Satz 3.5 kann in diesem Kapitel noch nicht vollständig geführt werden. Wir benötigen dafür Satz 5.9 aus Kapitel 5 sowie die Sätze 3.6 und 3.7. In Satz 5.10 wird der Beweis dann zum Abschluß gebracht.

Satz 3.6 Es sei L eine deterministische Typ-2-Sprache und R eine reguläre Sprache. Dann ist auch $L \cap R$ eine deterministische Typ-2-Sprache.

Beweis: Es sei $K = (V, F)$ ein deterministischer Kellerautomat mit $L = L(K)$ und $\bar{E} = (\bar{V}, \bar{F})$ ein endlicher erkennender Automat mit $R = L(\bar{E})$. Es sei S das Zustandsalphabet von K und $\bar{S}$ dasjenige von $\bar{E}$. Ohne Beschränkung der Allgemeinheit gelte $V_I = \bar{V}_T$. Es werde $V' = V_I = \bar{V}_T$ gesetzt.

Diese Forderung ist zulässig, denn wir können bei diesen deterministischen Automaten das Eingabealphabet beliebig erweitern, ohne die akzeptierte Sprache zu verändern. Dies geschieht durch die Einführung eines „Papierkorbzustandes" p. Im Falle eines endlichen erkennenden Automaten wird für alle neuen Eingabesymbole $a \in V'$ und alle $s \in \bar{S}$ eine Produktion (sa, p) eingeführt. Außerdem wird (pb, p) für alle $b \in V'$ gesetzt. Bei Kellerautomaten wird für alle neuen Eingabesymbole $a \in V'$ und alle $z \in V_Z$, $s_i \in S$ eine neue Produktion (zs_ia, zp) definiert. Weiter wird für alle $b \in V'$ die Produktion (zpb, zp) gegeben. In beiden Fällen erhalten wir deterministische Automaten, die dieselben Sprachen wie die gegebenen Automaten akzeptieren.

Wir konstruieren einen deterministischen Kellerautomaten $\tilde{K} = (\tilde{V}, \tilde{F})$ wie folgt. Es sei $\tilde{S} = S \times \bar{S}$ sein Zustandsalphabet, $\tilde{S}_1 = S_1 \times \bar{S}_1$ sei die Menge der Endzustände und $\tilde{s}_0 = (s_0, \bar{s}_0)$ der Anfangszustand. Es sei V' das Eingabealphabet. Das Kelleralphabet V_Z und das Startzeichen z_0 seien wie in K definiert. Wir setzen weiter

$$\begin{aligned}\tilde{F} = \{&(z(s_i, \bar{s}_l), Q(s_j, \bar{s}_l)) \mid (zs_i, Qs_j) \in F, \bar{s}_l \in \bar{S}\} \\ &\cup \{(z(s_i, \bar{s}_l)a, Q(s_j, \bar{s}_m)) \mid (zs_ia, Qs_j) \in F, (\bar{s}_la, \bar{s}_m) \in \bar{F}\}.\end{aligned}$$

Die erste Teilmenge ist anschaulich so zu verstehen, daß die $\bar{E}$-Komponente des Kellerautomaten nicht arbeitet, wenn seine K-Komponente kein Eingabesymbol verarbeitet. Ersichtlich ist $\tilde{K}$ ein deterministischer Kellerautomat.

Wir zeigen

$$\left(z_0(s_0, \bar{s}_0)w \Longrightarrow_{\tilde{K}}^{k} Q(s, \bar{s})\right) \iff \left((z_0s_0w \Longrightarrow_K^k Qs) \wedge (\bar{s}_0w \Longrightarrow_{\bar{E}}^{|w|} \bar{s})\right).$$

Dabei bedeutet $\Longrightarrow^k$ eine Ableitung in k Schritten. Der Beweis erfolgt durch Induktion über die Länge k der Ableitung des Kellerautomaten $\tilde{K}$. Für $k = 0$ erhalten wir die Aussage mit $w = \varepsilon$, $s_0 = s$, $\bar{s}_0 = \bar{s}$ und $z_0 = Q$. Wir nehmen an, daß die Aussage für $k - 1$ erfüllt ist. Es sei $w = w_1a$ mit $a = \varepsilon$ oder $a \in V'$. Es gilt nun

$$\begin{aligned}&\left(z_0(s_0, \bar{s}_0)w_1a \Longrightarrow_{\tilde{K}}^{k-1} Q'(s', \bar{s}')a \Longrightarrow_{\tilde{K}} Q(s, \bar{s})\right) \\ \iff &\left((z_0s_0w_1 \Longrightarrow_K^{k-1} Q's') \wedge (\bar{s}_ow_1 \Longrightarrow_{\bar{E}}^{|w_1|} \bar{s}') \wedge (Q's'a \Longrightarrow_K Qs) \wedge (\bar{s}'a \Longrightarrow_{\bar{E}}^{|a|} \bar{s})\right) \\ \iff &\left((z_0s_0w \Longrightarrow_K^k Qs) \wedge (\bar{s}_0w \Longrightarrow_{\bar{E}}^{|w|} \bar{s})\right).\end{aligned}$$

Da der Keller niemals leer wird, gilt immer $Q' \neq \varepsilon$. Aufgrund der Äquivalenz folgt

$$w \in L(\tilde{K}) \iff s \in S_1, \bar{s} \in \bar{S}_1 \iff w \in L(K), w \in L(\bar{E}) \iff w \in L \cap R. \quad \square$$

Der Beweis gilt entsprechend auch für beliebige Kellerautomaten, so daß die Abschlußeigenschaft des Satzes unter Verwendung von Satz 3.4 auch für beliebige Typ-2-Sprachen gilt. Wir werden dieses Ergebnis mit Hilfe von Grammatiken in Satz 5.13 beweisen.

Definition 3.9 Es sei $w = a_1 \dots a_n$ mit $a_i \in V_T$, $i = 1, \dots, n$, $n \in \mathbb{N}$. Dann sind $\varepsilon, a_1, a_1a_2, \dots, a_1 \dots a_n$ alle *Präfixe* von w. Für eine Sprache L und $i \in \mathbb{N}$ sei

$$P_i(L) = \{w \mid w \in L, \text{ genau } i-1 \text{ Präfixe von } w \text{ gehören zu } L\}.$$

Für $n \in \mathbb{N}$ verwenden wir die Bezeichnung $\underline{n} = \{0, 1, \dots, n\}$. □

Satz 3.7 Es sei L eine deterministische Typ-2-Sprache. Dann ist $P_k(L)$ für alle $k \in \mathbb{N}$ eine deterministische Typ-2-Sprache.

Beweis: Es sei K ein deterministischer Kellerautomat mit $L = L(K)$, und es sei $k \in \mathbb{N}$. Wir konstruieren einen deterministischen Kellerautomaten K_k wie folgt. Es sei $S \times \underline{k} \times \underline{1}$ das Zustandsalphabet, wobei $S_1 \times \{k-1\} \times \underline{1}$ die Menge der Endzustände und $(s_0, 1, 1)$ für $s_0 \in S_1$ oder $k = 1$ bzw. $(s_0, 0, 0)$ für $s_0 \notin S_1$ und $k \neq 1$ der Anfangszustand ist. Der Keller ist wie in K definiert. Die Produktionen aus F_k sind wie folgt gegeben:

$$(z(s_i, \kappa, p), Q(s_j, \kappa', q)) \in F_k : \Longleftrightarrow (zs_i, Qs_j) \in F \wedge$$
$$q = \begin{cases} 1 \text{ für } s_j \in S_1 \\ p \text{ für } s_j \notin S_1 \end{cases} \wedge \kappa' = \begin{cases} \min\{k, \kappa+1\} & \text{für } s_j \in S_1 \wedge p = 0 \\ \kappa & \text{für } s_j \notin S_1 \vee p = 1 \end{cases},$$

$$(z(s_i, \kappa, p)a, Q(s_j, \kappa', q)) \in F_k : \Longleftrightarrow (zs_1a, Qs_j) \in F \wedge$$
$$q = \begin{cases} 1 \text{ für } s_j \in S_1 \\ 0 \text{ für } s_j \notin S_1 \end{cases} \wedge \kappa' = \begin{cases} \min\{k, \kappa+1\} & \text{für } s_j \in S_1 \\ \kappa & \text{für } s_j \notin S_1 \end{cases}.$$

Offenbar ist K_k ein deterministischer Kellerautomat. Die erste Komponente der Zustandsmenge führt die Arbeit von K aus, die zweite Komponente zählt, wieviele Präfixe von w in L enthalten sind. Die Zahl in $\underline{k}$ wird genau dann bei Abarbeitung von w um 1 erhöht, wenn das entsprechende Präfix von w in L liegt, also das Wort von Beginn des Eingabebandes bis einschließlich dem Zeichen unmittelbar links von dem Symbol, über dem der Kopf nach Ausführung des aktuellen Schrittes steht. Dabei ist die dritte Komponente notwendig, um falsches Zählen bei den Produktionen $(z(s_i, \kappa, p), Q(s_j, \kappa', q))$ zu verhindern. Bei einem entsprechenden Schritt des Kellerautomaten wird ja das Eingabesymbol nicht berücksichtigt, und der Kopf bewegt sich nicht. Man erhält nur dann ein Präfix von w als ein weiteres Wort aus L, wenn es nicht vorher bereits gezählt wurde. Daß ein Präfix gezählt wurde, wird sich durch Setzen von p auf 1 gemerkt. Hat man mehr als k Präfixe von L gezählt, dann hat die zweite Komponente den Wert k und man gelangt nie in einen Endzustand. Für $s_0 \in S_1$ muß im Anfangszustand $(s_0, 1, 1)$ gestartet werden, da dann das leere Wort mitgezählt werden muß. Ist $k = 1$, so gilt $P_1(L) = \emptyset$, und mit demselben Anfangszustand gelangt man nie in einen Endzustand. In den anderen Fällen beginnt das Zählen normal mit Hilfe des Anfangszustands $(s_0, 0, 0)$. Da K_k deterministisch ist, sind alle diese Feststellungen eindeutig. Wir erhalten so offenbar $L(K_k) = P_k(L)$. □

Satz 3.8 Es gilt $\mathfrak{L}_3 \subsetneq \mathfrak{L}_{2,d}$.

Beweis: Es sei $L \in \mathfrak{L}_3$. Dann existiert nach Satz 3.3 ein endlicher erkennender Automat E mit $L(E) = L$. Wir können E als einen deterministischen Kellerautomaten mit

genau einem Kellersymbol z_0 auffassen, bei dem der Keller niemals verändert wird. Somit gilt $\mathfrak{L}_3 \subset \mathfrak{L}_{2,d}$.

Wir zeigen, daß die Sprache $L_1 = \{a^n b a^n \mid n \in \mathbb{N}\}$ durch einen deterministischen Kellerautomaten K akzeptiert wird. Wir skizzieren den Automaten K. Der Kopf wird mit jedem Schritt bewegt. Falls zuerst ein Symbol b gelesen wird, geht K in einen Fehlerzustand f über, aus dem man nie in einen Endzustand gelangt. Anderenfalls wird das gelesene a als a' in den Keller geschrieben. Weitere gelesene a werden als a auf den bisherigen Kellerinhalt gestapelt. Wird dann erstmals b gelesen, erfolgt ein Übergang in einen entsprechenden Zustand s, der die Überprüfung der restlichen Symbole vornimmt. Wird jetzt a gelesen und ist a das oberste Kellerelement, so wird es im Keller gelöscht. Ist dabei das oberste Kellerelement a', so geht K in den Endzustand s_1 über. In allen anderen Fällen, also auch, wenn im Endzustand noch ein weiteres Symbol gelesen wird, gelangt K in den Fehlerzustand f. Ersichtlich gilt $L(K) = L_1$.

Schließlich nehmen wir an, daß L_1 durch einen endlichen erkennenden Automaten E akzeptiert wird. Wir wählen ein spezielles $n > \text{card}(S)$ und betrachten damit das Wort $a^n b a^n \in L_1$. Dann gilt

$$s_0 a^n b a^n \Longrightarrow_E^* s_1 \in S_1.$$

Wegen $n > \text{card}(S)$ existieren $s \in S$ und $p, q \in \mathbb{N}$, $1 \leq p < q \leq n$, mit

$$s_0 a^p \Longrightarrow_E^* s, \; s_0 a^q \Longrightarrow_E^* s.$$

Da der Zustand s nach $(q-p)$-facher Eingabe von a wiederholt wird, kann er entsprechend auch beliebig oft wiederholt werden. Es folgt also

$$s_0 a^p (a^{q-p})^\nu a^{n-q} b a^n \Longrightarrow_E^* s_1 \in S_1 \text{ für alle } \nu \in \mathbb{N}_0$$

und damit $a^p (a^{q-p})^\nu a^{n-q} b a^n \in L_1$ für alle $\nu \in \mathbb{N}_0$, was im Widerspruch zur Gestalt von L_1 steht. L_1 ist also keine Typ-3-Sprache. □

3.4 Linear beschränkte Automaten und Typ-1-Sprachen

Während bei Kellerautomaten potentiell unendlich viel Speicher in Form des Kellerbandes zur Verfügung steht, der allerdings nur in eingeschränkter Weise erreichbar ist, sind die Verhältnisse bei linear beschränkten Automaten anders. Hier ist der zur Verfügung stehende Speicherplatz nur ein lineares Vielfaches der Anzahl der Zeichen eines vorgegebenen Wortes. Auf alle Speicherfelder kann dagegen beliebig oft zugegriffen werden.

Definition 3.10 Es sei $B = (V, F)$ ein kombinatorisches System. B heißt *linear beschränkter Automat*, wenn die folgenden Aussagen erfüllt sind:

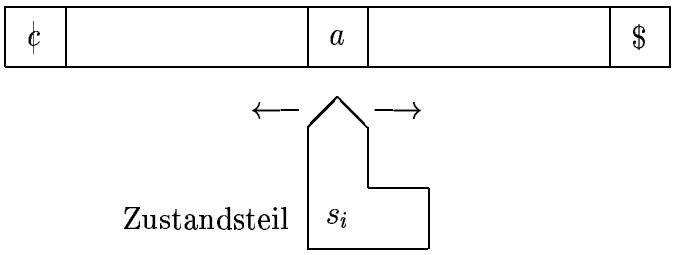
(a) Es gilt $V = S \cup V_T$ mit *Zustandsalphabet* S und *Bandalphabet* V_T, wobei $S \cap V_T = \emptyset$ gilt. Eine Teilmenge $V_I \subset V_T$ wird als *Eingabealphabet* ausgezeichnet.

(b) Es werden ein *Anfangszustand* $s_0 \in S$, eine *Endzustandsmenge* $S_1 \subset S$, ein *linker Randmarkierer* $\not\in V_T$ und ein *rechter Randmarkierer* $\$ \in V_T$ ausgezeichnet.

(c) Die Produktionen von F sind von einer der folgenden Formen:
- (α) $s_i a \to s_j b$ mit $s_i, s_j \in S$, $a, b \in V_T - \{\not\in, \$\}$ (Überschreiben),
- (β) $s_i a \to a s_j$ mit $s_i, s_j \in S$, $a \in V_T - \{\$\}$ (Rechtsbewegung),
- (γ) $c s_i a \to s_j c a$ mit $s_i, s_j \in S$, $a \in V_T - \{\not\in\}$, $c \in V_T - \{\$\}$ (Linksbewegung),
- (δ) $s_i \not\in \to s_j \not\in$ und $s_i \$ \to s_j \$$ für $s_i, s_j \in S$ (Zustandsänderungen an den Rändern).

Für jedes $(s_i, s_j, a) \in S \times S \times (V_T - \{\not\in\})$ enthält F entweder keine Produktion (γ) oder aber für jedes $c \in V_T - \{\$\}$ eine Produktion (γ). □

Wir können einen linear beschränkten Automaten durch das folgende Bild veranschaulichen:



Ein linear beschränkter Automat besitzt ein beschränktes Lese- und Schreibband sowie einen Zustandsteil. Zwischen den beiden Randsymbolen wird das zu bearbeitende Wort w eingetragen, das weder $\not\in$ noch \$ enthält. Der Automat geht in Abhängigkeit von seinem Zustand s_i und dem gerade gelesenen Symbol a in einen Zustand s_j über und ändert entweder gemäß (α) das gelesene Zeichen oder bewegt seinen Schreib- und Lesekopf einen Schritt nach rechts (β) oder links (γ). Die Randsymbole dürfen nicht überschrieben werden, jedoch kann dort eine Zustandsänderung stattfinden (δ). Eine Bewegung über die Ränder hinaus ist nicht möglich.

Definition 3.11 Es sei B ein linear beschränkter Automat.

(a) Es sei $s_i \in S$, $Q \in V_T^*$. Das Wort $s_i Q\$$ heißt *final*, wenn $Q\$$ mit keinem Symbol $a \in V_T$ beginnt, für das $s_i a$ Teilwort der linken Seite einer Produktion in F ist.

(b) Die Menge

$$L(B) = \{w \mid w \in V_I^*,\ \not\in s_o w\$ \Longrightarrow^* \not\in Q_1 s_1 Q_2\$, Q_1, Q_2 \in V_T^*, s_1 \in S_1,\ s_1 Q_2\$ \text{ final}\}$$

heißt die *von B angenommene* oder *erkannte (akzeptierte)* Sprache.

(c) Es sei $L \subset V_I^*$ eine Sprache. L heißt *von einem linear beschränkten Automaten erkennbar (akzeptierbar)*, wenn ein linear beschränkter Automat B mit $L = L(B)$ existiert. □

Ein Wort $w \in V_I^+$ wird von einem linear beschränkten Automaten akzeptiert, wenn er, ausgehend vom Anfangszustand und angesetzt auf das erste Zeichen von w,

irgendwann in einen Endzustand gelangt und hält, d.h., wenn er einen Endzustand erreicht, bei dem keine Produktion mehr anwendbar ist. Beim leeren Wort ε steht der Kopf des Automaten zu Beginn auf dem rechten Rand.

Satz 3.9 $\mathfrak{L}_1$ ist gleich der Familie der durch nichtdeterministische linear beschränkte Automaten akzeptierten Sprachen.

Beweis: Wir gehen von einer beliebigen Typ-1-Grammatik $G = (V_N, V_I, X_0, F)$ aus und skizzieren die Konstruktion eines linear beschränkten Automaten B mit $L = L(B) = L(G)$. Falls $\varepsilon \in L$ gilt, was nach Definition 2.6 mit der Existenz der Produktion $X_0 \to \varepsilon$ in G äquivalent ist, dann kann B immer so konstruiert werden, daß $s_0 \in S_1$ ist mit $s_0\$$ final. Offenbar erhalten wir so $\varepsilon \in L(B)$. Für ein beliebiges $w = a_1 \ldots a_n \in V_I^+$ können wir durch einen Vor- und Rücklauf des Kopfes eine Unterteilung des Eingabebandes von B in zwei Spuren erreichen:

¢	a_1	a_2	a_3	$\cdots$	a_n	\$	Spur 1
	b	b	b	$\cdots$	b		Spur 2

Dabei enthält die erste Spur das Eingabewort $w = a_1 \ldots a_n$, während die Felder von Spur 2 jeweils mit einem „Blankzeichen“ $b \notin V_I$ beschrieben werden. Während der weiteren Arbeit wird die Spur 1 nicht verändert, während auf der zweiten Spur der Erzeugungsprozeß der Grammatik G simuliert wird. Zuerst wird X_0 in das am weitesten links stehende Feld der Spur 2 geschrieben. Wir nehmen nun allgemein an, daß links auf der Spur 2 ein Wort steht, das gemäß G erzeugbar ist. Nach rechts sei das Band durch Zeichen b aufgefüllt. Es wird zunächst eine Produktion $Q_1XQ_2 \to Q_1RQ_2$ der Grammatik nichtdeterministisch gewählt. Diese Wahl kann sich der linear beschränkte Automat durch einen Übergang in einen entsprechenden Zustand merken. Danach sucht B nichtdeterministisch das Wort Q_1XQ_2 auf Spur 2, wofür dieser Produktion entsprechende Zustände und Produktionen von B notwendig sind. Wird das Wort nicht gefunden, hält B in einem Nichtendzustand. Anderenfalls wird X durch R ersetzt, wobei die eventuell rechts von X stehenden Symbole um $|R| - 1$ Felder nach rechts verschoben werden müssen. Die Verschiebung um eine Stelle nach rechts kann so durchgeführt werden, daß der Kopf des linear beschränkten Automaten nach rechts läuft, wiederholt das gelesene Symbol in den Zustand aufnimmt (im Zustand kann eine Komponente für Bandsymbole vorhanden sein) und dieses durch das zuvor in den Zustand aufgenommene Zeichen ersetzt. Entsprechend können größere Blöcke von Wörtern verschoben werden. Würde bei diesem Verschieben die Spur 2 nach rechts verlassen, so hält statt dessen der Automat B ohne Akzeptierung. Da bei einer Typ-1-Grammatik die rechte Seite einer Produktion größer oder gleich ihrer linken Seite ist, kann in diesem Fall das Wort w nicht mehr erzeugt werden. Ist jedoch die Anwendung der Produktion erfolgreich simuliert worden, so wird nichtdeterministisch eine weitere Produktion simuliert oder der Erzeugungsprozeß beendet. Im letzten Fall vergleicht der Automat B die Inschriften beider Spuren. Stimmen sie überein, so hält B in einem Endzustand, anderenfalls in einem Nichtendzustand. Offenbar gilt $L(B) = L(G)$.

Für die umgekehrte Beweisrichtung gehen wir von einem beliebigen linear beschränkten Automaten $B = (V, F)$ aus und konstruieren eine monotone Grammatik $G = (V_N, V_I, X_0, F')$ mit $L(G) = L(B)$. Nach Satz 2.7 existiert dann auch eine äquivalente Typ-1-Grammatik. Ohne Beschränkung der Allgemeinheit können wir annehmen, daß ein Wort genau dann von B akzeptiert wird, wenn B auf dem rechten Rand in einem Endzustand stehen bleibt. Ist dies nicht von vornherein der Fall, so werden mit einem neuen Endzustand $\bar{s}_1 \notin S$ neue Produktionen $\bar{s}_1 a \to a\bar{s}_1$ für alle $a \in V_T - \{\$\}$ zur Produktionenmenge hinzugefügt, außerdem auch noch für alle $s \in S_1$, $b \in V_T - \{\$\}$, für die sb final ist, Produktionen $sb \to \bar{s}_1 b$. Der so geänderte Automat erzeugt dieselbe Sprache und besitzt die gewünschte Eigenschaft. Wir geben die Produktionenmenge F' der Grammatik G und damit implizit auch die Menge V_N an. Es gilt $\varepsilon \in L(B)$ genau dann, wenn $s_0 \in S_1$ erfüllt und $s_0\$$ final ist. Falls $\varepsilon \in L(B)$ gilt, wird die Produktion $X_0 \to \varepsilon$ aufgenommen, wobei X_0 das Anfangssymbol der Grammatik ist. In jedem Fall erhalten wir die „Anfangsproduktionen“

$$X_0 \to A_1$$

und

$$A_1 \to [a, ¢s_0a]A_2,\ A_1 \to [a, ¢s_0a\$],\ A_2 \to [a, a]A_2,\ A_2 \to [a, a\$]$$

für alle $a \in V_I$. Auf der ersten Komponente wird das terminale Wort gebildet, auf der zweiten Komponente wird das Band des linear beschränkten Automaten simuliert. Die erste Komponente bleibt unverändert, bis die „Schlußproduktionen“ angewendet werden. Für ein Wort $w = a_1 \dots a_n \in V_I^+$, $a_i \in V_I$, $i = 1, \dots, n$, $n \in \mathbb{N}$, erhält man eine Ableitung

$$X_0 \Longrightarrow_G A_1 \Longrightarrow_G [a_1, ¢s_0a_1]A_2 \Longrightarrow_G^* [a_1, ¢s_0a_1][a_2, a_2] \dots [a_n, a_n\$] \text{ für } n \geq 2$$

oder

$$X_0 \Longrightarrow_G A_1 \Longrightarrow_G [a_1, ¢s_0a_1\$] \text{ für } n = 1.$$

Wir kommen nun zu den „Simulationsproduktionen“. Es bedeute $(¢)$, daß das Symbol ¢ auf der linken und rechten Seite einer Produktion entweder gleichzeitig vorkommt und auch gleichzeitig fehlt. Beide Möglichkeiten sind in der Produktionenmenge zu berücksichtigen. Entsprechendes gilt für $(\$)$. Für alle $a, d, d_1, d_2 \in V_I$ erhalten wir die folgenden Produktionen:

(1) $[d, (¢)s_ia_1(\$)] \to [d, (¢)s_ja_2(\$)]$ für $(s_ia_1, s_ja_2) \in F$,

(2) $[d_1, (¢)s_ia_1][d_2, a_2(\$)] \to [d_1, (¢)a_1][d_2, s_ja_2(\$)]$ für $(s_ia_1, a_1s_j) \in F$,
$[d, s_i¢a(\$)] \to [d, ¢s_ja(\$)]$ für $(s_i¢, ¢s_j) \in F$,
$[d, (¢)s_ia_1\$] \to [d, (¢)a_1s_j\$]$ für $(s_ia_1, a_1s_j) \in F$ (am rechten Rand),

(3) $[d_1, (¢)a_1][d_2, s_ia_2(\$)] \to [d_1, (¢)s_ja_1][d_2, a_2(\$)]$ für $(a_1s_ia_2, s_ja_1a_2) \in F$,
$[d, ¢s_ia(\$)] \to [d, s_j¢a(\$)]$ für $(¢s_ia, s_j¢a) \in F$,

(4) $[d, s_i¢a(\$)] \to [d, s_j¢a(\$)]$ für $(s_i¢, s_j¢) \in F$ und
$[d, (¢)as_i\$] \to [d, (¢)as_j\$]$ für $(s_i\$, s_j\$) \in F$.

Falls eine Ableitung

$$¢s_0a_1 \ldots a_n\$ \Longrightarrow_B^* ¢Qs_1\$ \text{ mit } s_1 \in S_1, Q = b_1 \ldots b_n \in V_T^+$$

gemäß B existiert, so gibt es offenbar eine Ableitung

$$[a_1, ¢s_0a_1][a_2, a_2] \ldots [a_n, a_n\$] \Longrightarrow_G^* [a_1, ¢b_1][a_2, b_2] \ldots [a_n, b_ns_1\$] \text{ für } n \geq 2$$

oder

$$[a_1, ¢s_0a_1\$] \Longrightarrow_G^* [a_1, ¢b_1s_1\$] \text{ für } n = 1$$

gemäß G.

Die „Schlußproduktionen“ sind

$$[a, (¢)bs\$] \to a \text{ für alle } s \in S_1 \text{ mit } s\$ \text{ final}, a \in V_I, b \in V_T - \{¢, \$\}$$

und

$$[a, (¢)b_1]b \to ab \text{ für alle } a, b \in V_I, b_1 \in V_T - \{¢, \$\}.$$

Für einen Endzustand $s_1 \in S_1$ mit $s_1\$$ final folgt dann

$$[a_1, ¢b_1] \ldots [a_n, b_ns_1\$] \Longrightarrow_G^* a_1 \ldots a_n.$$

Da $a_1 \ldots a_n \in L(B)$ ist, folgt $L(B) \subset L(G)$. Umgekehrt sind die Produktionen so konstruiert, daß nur solche Ableitungen wie oben zu Wörtern über V_I führen können. Dabei ist es allerdings möglich, daß schon Simulationsschritte durchgeführt werden, bevor A_2 durch eine Produktion $A_2 \to [a, a\$]$ entfernt wird. Eine Anwendung einer „Schlußproduktion“ ist jedoch erst nach einer solchen Löschung von A_2 möglich. Es gilt also $L(B) = L(G)$. Wir erkennen außerdem, daß G monoton ist. □

Definition 3.12 Es sei $B = (V, F)$ ein linear beschränkter Automat. B heißt *deterministischer linear beschränkter Automat*, wenn für jedes Paar $(s_i, a) \in S \times V_T$ genau eine der folgenden Bedingungen erfüllt ist:

(a) Es existiert genau eine Produktion der Form (α) in F.
(b) Es existiert genau eine Produktion der Form (β) in F.
(c) Es existiert genau eine Produktion der Form (δ) in F.
(d) Es existiert genau eine Menge von Produktionen der Form (γ) in F, wobei c alle Werte aus $V_T - \{\$\}$ annimmt.
(e) Es existiert keine Produktion für (s_i, a) in F. □

Durch diese Forderungen ist die eindeutige Reaktion des deterministischen linear beschränkten Automaten für das Paar (s_i, a) gewährleistet. Im Fall (e) hält der Automat. Wir geben jetzt ein Beispiel für einen deterministischen linear beschränkten Automaten an.

Beispiel 3.1 Ein deterministischer linear beschränkter Automat, der die Sprache

$$\{a^n b^n c^n \mid n \in \mathbb{N}\}$$

erkennt, arbeitet nach folgendem Prinzip: Durch Hin- und Herlaufen und Markieren von je einem a, b und c pro Durchgang wird geprüft, ob gleich viele a's, b's und c's vorhanden sind. Wir verzichten auf die explizite Angabe der Produktionen, die etwas aufwendig ist. □

Abschließend formulieren wir das folgende bekannte offene Problem der Automatentheorie, das sogenannte *lba-Problem*:

Offenes Problem Gibt es zu jedem linear beschränkten Automaten B einen deterministischen linear beschränkten Automaten B_d mit $L(B_d) = L(B)$?

Falls diese Frage negativ beantwortet wird oder auch solange dieses Problem noch nicht geklärt ist, ist die folgende Aussage sinnvoll und von Bedeutung:

> Falls L eine Typ-2-Sprache ist, dann existiert ein deterministischer linear beschränkter Automat B mit $L(B) = L$.

Eine Beweisskizze für diese Aussage werden wir in Satz 5.7 liefern.

3.5 Turingmaschinen und Typ-0-Sprachen

Definition 3.13 Es sei $T = (V, F)$ ein kombinatorisches System. T heißt *Turingmaschine*, wenn die folgenden Eigenschaften erfüllt sind:

(a) Es gilt $V = S \cup V_T$ mit *Zustandsalphabet* S und *Bandalphabet* V_T, wobei $S \cap V_T = \emptyset$ gilt.

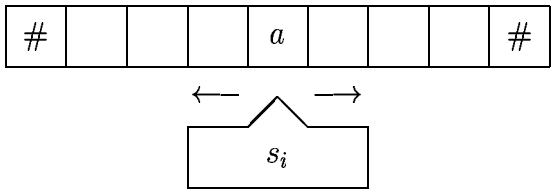
(b) Es wird ein *Anfangszustand* $s_0 \in S$, *ein Grenzzeichen* $\# \in V_T$ und eine *Menge von Endzuständen* $S_1 \subset S$ ausgezeichnet. Weiter sei $V_1 = V_T - \{\#\} \neq \emptyset$. Es wird ein *Blankzeichen* $B \in V_1$ und eine *Eingabemenge* $V_I \subset V_1 - \{B\}$ ausgezeichnet.

(c) Die Produktionen von F haben die folgende Gestalt:
- (α) $(s_i a, s_j b)$ (Überschreiben),
- (β) $(s_i ac, a s_j c)$ (Rechtsbewegung),
- (γ) $(s_i a\#, a s_j B\#)$ (Rechtsbewegung und Bandverlängerung),
- (δ) $(c s_i a, s_j c a)$ (Linksbewegung),
- (ϵ) $(\# s_i a, \# s_j B a)$ (Linksbewegung und Bandverlängerung),

wobei $s_i, s_j \in S$, $a, b, c \in V_1$ gilt. Für jedes $(s_i, s_j, a) \in S \times S \times V_1$ enthält F entweder keine Produktion der Form (β) und (γ) (bzw. (δ) und (ϵ)) oder aber F enthält für alle $c \in V_1$ sowohl die Produktionen der Form (β) als auch (γ) (bzw. (δ) als auch (ϵ)).

Falls $s_i a$ für kein $s_i \in S$, $a \in V_1$ Teilwort der linken Seite von mindestens zwei Produktionen der Form (α), (γ) und (ϵ) ist, so heißt T *deterministische Turingmaschine*. □

Man kann eine Turingmaschine interpretieren als ein Gerät, das einen Lese- und Schreibkopf hat sowie ein Band, das nach beiden Seiten potentiell unendlich ist. Diese Unendlichkeit wird durch die Möglichkeit der Bandverlängerung durch die Produktionen (γ) und (ϵ) gegeben. Der Lese- und Schreibkopf steht jeweils über einem Feld des Bandes, das mit einem Symbol $a \in V_1$ beschrieben ist.



Befindet sich dabei die Turingmaschine im Zustand s_i, so geht die Turingmaschine in einen neuen Zustand s_j über, und es wird entweder das Feld durch ein Symbol $b \in V_1$ überschrieben, was durch eine Produktion (α) ausgedrückt wird, oder es findet eine Links- oder Rechtsbewegung statt. Die Links- bzw. Rechtsbewegungen sind nur abhängig vom gerade gelesenen Symbol, sie erfordern jedoch in unserer Definition den linken bzw. rechten Kontext, d.h. eine Anzahl von $\mathrm{card}(V_T)$ Produktionen für jedes $(s_i, a) \in S \times V_1$. Eine Linksbewegung läßt sich, ebenso wie bei linear beschränkten Automaten, durch solche Produktionen nicht anders darstellen. Für eine Rechtsbewegung ist der rechte Kontext wegen einer eventuellen Bandverlängerung nötig. Das Grenzzeichen wird bei diesen Bewegungen schon erkannt, wenn die Turingmaschine ein Feld daneben steht. Das Grenzzeichen kann niemals direkt gelesen werden.

Die Definition einer Turingmaschine als kombinatorisches System erfordert also einen gewissen zusätzlichen Definitionsaufwand. Meistens werden Turingmaschinen mit Hilfe von Abbildungen kürzer definiert (siehe z.B. [30], Definition 3.1). Der Vorteil von Definition 3.13 ist, daß damit die Arbeit der Turingmaschine durch die Ableitungen des kombinatorischen Systems beschrieben werden kann. Dafür ist dann auch das Grenzzeichen notwendig, auf das wir sonst bei entsprechenden Vereinfachungen in Definition 3.13 verzichten könnten.

Definition 3.14 Es sei T eine Turingmaschine.

(a) Es sei $s_i \in S$, $Q \in V_T^*$. Das Wort s_iQ heißt *final*, wenn Q mit keinem Symbol $a \in V_T$ beginnt, für das s_ia Teilwort der linken Seite einer Produktion in F ist.

(b) Es heißt

$$\begin{aligned} L(T) \;=\; \{w \mid\; & w \in V_I^*,\ \#s_o w'\# \Longrightarrow^* \#Q_1 s_1 Q_2 \#,\ Q_1 \in V_1^*, Q_2 \in V_1^+, \\ & s_1 \in S_1, s_1Q_2 \text{ final},\ w' = w \text{ für } w \neq \varepsilon,\ w' = B \text{ für } w = \varepsilon\} \end{aligned}$$

die *von T angenommene* oder *erkannte (akzeptierte)* Sprache.

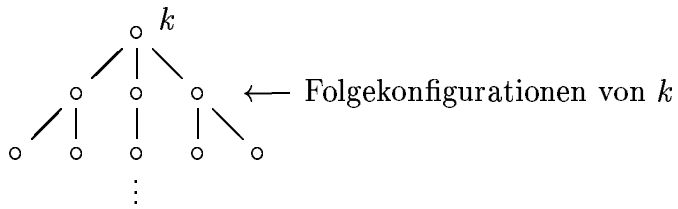
(c) Es sei $L \subset V_I^*$ eine Sprache. L heißt *von einer Turingmaschine erkennbar (akzeptierbar)*, wenn eine Turingmaschine T mit $L = L(T)$ existiert. □

Die Turingmaschine T beginnt also ihre Arbeit im Anfangszustand, wobei in den Feldern des Bandes die Zeichen des Wortes w innerhalb der Grenzzeichen eingetragen sind. Der Kopf der Maschine steht, falls $w \neq \varepsilon$ gilt, über dem Feld mit dem ersten Symbol von w, anderenfalls über dem Blankzeichen B. Für $w \neq \varepsilon$ könnte auch $w' = wB$ in Definition 3.14(b) eingesetzt werden, ohne daß die erkannte Sprache verändert würde.

Man könnte vermuten, daß nichtdeterministische Turingmaschinen mehr leisten als deterministische. Das ist jedoch nicht der Fall.

Satz 3.10 Deterministische und nichtdeterministische Turingmaschinen akzeptieren dieselbe Sprachfamilie.

Beweis: Wir geben nur eine Skizze für den im Detail komplizierten Beweis an. Die nichtdeterministische Arbeitsweise einer nichtdeterministischen Turingmaschine ist durch einen Baum der Art



darstellbar. Unter einer Konfiguration verstehen wir den Zustand, die Beschreibung des Bandes und die Position des Kopfes der Turingmaschine zu einem bestimmten Zeitpunkt. Zu einer Konfiguration dieser Maschine sind, im Gegensatz zum Fall einer deterministischen Turingmaschine, mehrere Folgekonfigurationen möglich. Eine Simulierung durch eine deterministische Turingmaschine erreicht man, indem man „Schicht für Schicht“ alle Folgekonfigurationen verfolgt. Dabei werden auf dem Band der deterministischen Turingmaschine alle Konfigurationen der nichtdeterministischen Turingmaschine abgespeichert, die noch nicht weiterverfolgt wurden. Die Einzelheiten dieser Konstruktion sind sehr aufwendig und sollen hier nicht dargestellt werden. Eine deterministische Turingmaschine kann natürlich als spezielle nichtdeterministische Turingmaschine aufgefaßt werden. □

Satz 3.11 $\mathfrak{L}_0$ ist gleich der Familie der durch nichtdeterministische Turingmaschinen akzeptierten Sprachen.

Beweis: Wir gehen von einer beliebigen Typ-0-Grammatik $G = (V_N, V_I, X_0, F)$ aus und skizzieren die Konstruktion einer Turingmaschine T mit $L = L(T) = L(G)$. Die Vorgehensweise entspricht im wesentlichen der Konstruktion des linear beschränkten Automaten im Beweis von Satz 3.9. Für ein beliebiges $w \in V_I^*$ können wir durch einen Vor- und Rücklauf des Kopfes eine Unterteilung des Eingabebandes von T in zwei Spuren erreichen:

#	B	a_1	a_2	$\cdots$	a_n	B	#	Spur 1
	B	B	B	$\cdots$	B	B		Spur 2

Dabei hält die erste Spur das Eingabewort $w = a_1 \ldots a_n$ mit $n \in \mathbb{N}_0$, während die Felder von Spur 2 jeweils mit dem Zeichen $B \notin V_I$ beschrieben werden. Während der weiteren Arbeit wird die Spur 1, bis auf eine eventuelle Bandverlängerung nach rechts durch weitere Zeichen B rechts von w, nicht verändert, wohingegen auf der zweiten Spur der Erzeugungsprozeß der Grammatik G simuliert wird. Das Zeichen (B, B) ist das eigentliche Blankzeichen der zu konstruierenden Turingmaschine. Zuerst wird X_0 in das unter a_1 stehende Feld der Spur 2 geschrieben. Wir nehmen nun allgemein an, daß mit diesem Feld beginnend auf der Spur 2 ein Wort steht, das gemäß G erzeugbar ist. Nach rechts sei das Band durch Zeichen B aufgefüllt. Es wird zunächst eine Produktion $P \to Q$ der Grammatik nichtdeterministisch gewählt. Diese Wahl kann sich die Turingmaschine durch einen Übergang in einen entsprechenden Zustand merken. Danach sucht T nichtdeterministisch das Wort P auf Spur 2, wofür dieser Produktion entsprechende Zustände und Produktionen von T notwendig sind. Wird das Wort nicht gefunden, hält T in einem Nichtendzustand. Anderenfalls wird P durch Q ersetzt, wobei die eventuell rechts von P stehenden Symbole um $|Q| - |P|$ Felder nach rechts (falls $|Q| > |P|$) bzw. um $|P| - |Q|$ Felder nach links (falls $|Q| < |P|$) verschoben werden müssen. Diese Verschiebungen können ähnlich wie bei linear beschränkten Automaten durchgeführt werden. Würde bei diesem Verschieben die Spur 2 nach rechts verlassen, so findet statt dessen eine Bandverlängerung statt. Bei einem Verschieben nach links werden die frei werdenden Felder durch das Zeichen B besetzt. Ist die Anwendung der Produktion erfolgreich simuliert worden, so wird nichtdeterministisch eine weitere Produktion simuliert oder der Erzeugungsprozeß beendet. Im letzten Fall vergleicht der Automat T die Inschriften beider Spuren. Stimmen sie überein, so hält T in einem Endzustand, anderenfalls in einem Nichtendzustand. Offenbar gilt $L(T) = L(G)$.

Für die umgekehrte Beweisrichtung betrachten wir eine Turingmaschine T mit $L = L(T)$. Aus T wird eine analysierende Typ-0-Grammatik $G = (V_N, V_T, X_0, F)$ konstruiert, die L erkennt. Wir setzen $V_N = (V - V_I) \cup \{X_0, X_1, X_2\}$, $V_T = V_I$, und die Produktionenmenge F wird durch die Produktionen von T gegeben sowie zusätzlich durch die Produktionen

$$
\begin{array}{ll}
(\varepsilon, \#s_0), & (\varepsilon, B\#), \\
(s_1 a, X_1) \text{ für alle } (s_1, a) \in S_1 \times V_1 \text{ mit } s_1 a \text{ final}, & \\
(X_1 b, X_1), & (bX_2, X_2) \text{ für alle } b \in V_1, \\
(X_1\#, X_2), & (\#X_2, X_0).
\end{array}
$$

Es sei $w \in L(T)$. Dann existiert gemäß G die Ableitung

$$
\begin{aligned}
w \Longrightarrow_G \#s_0 w \Longrightarrow_G \#s_0 wB\# \Longrightarrow_T^* \#Q_1 s_1 a Q_2\# \Longrightarrow_G \#Q_1 X_1 Q_2\# \\
\Longrightarrow_G^* \#Q_1 X_1\# \Longrightarrow_G \#Q_1 X_2 \Longrightarrow_G^* \#X_2 \Longrightarrow_G X_0.
\end{aligned}
$$

Man beachte dabei, daß in Definition 3.14(b) nach der Bemerkung im Anschluß an die Definition $w' = wB$ für alle w gesetzt werden kann. Wegen $w \in L(T)$ gilt $s_1 \in S_1$ mit $s_1 a$ final. Es folgt $w \in L(G)$.

Umgekehrt sei $w \in L(G)$. Aufgrund der Definition von F erfolgt die Ableitung $w \Longrightarrow_G^* X_0$ über die Schritte

$$w \Longrightarrow_G^* \#s_0 wB\# \Longrightarrow_T^* \#Q_1 s_1 a Q_2\# \Longrightarrow_G^* X_0$$

mit $s_1 a$ final. Dabei ist es auch möglich, daß $B\#$ am rechten Rand erst später eingefügt wird. Es können schon zuvor Schritte von T simuliert werden, die keine Rechtsbewegung über den rechten Rand hinaus veranlassen. Die Ableitungsschritte lassen sich jedoch ohne Änderung des erzeugten Wortes offensichtlich so vertauschen, daß sich die Ableitung wie angegeben ergibt. Dann gilt

$$\#s_0 wB\# \Longrightarrow_T^* \#Q_1 s_1 a Q_2\#$$

mit $s_1 a Q_2\#$ final. Es folgt $w \in L(T)$. □

Während die bisher besprochenen Automatentypen als spezielle Vorrichtungen zur Erkennung gewisser Sprachklassen geschaffen wurden, haben Turingmaschinen einen anderen Ausgangspunkt. Sie wurden 1936 von *A. Turing* als ein Modell vorgeschlagen, das den Begriff der Berechenbarkeit erklären soll, also als eine arithmetische Vorrichtung. Da der Begriff der Berechenbarkeit formal nicht faßbar ist, ist es mathematisch nicht beweisbar, daß Turingmaschinen eine adäquate Beschreibungsform der Berechenbarkeit darstellen. Es wird jedoch weitgehend die Churchsche These akzeptiert.

Churchsche These Eine partielle Funktion ist genau dann intuitiv berechenbar, wenn sie durch eine Turingmaschine berechnet wird. □

In der hier angegebenen Form wird die These auch als *Turingsche These* bezeichnet, wohingegen die Churchsche These ursprünglich für μ-rekursive Funktionen formuliert wurde. Da viele zu den Turingmaschinen äquivalente Berechnungsmodelle gefunden wurden (μ-rekursive Funktionen, Markov-Algorithmen (siehe Definition 2.2), applikative Algorithmen, imperative Algorithmen, Postsche Normalsysteme,...), kann man dies als einen Hinweis auf die Richtigkeit der Churchschen These verstehen.

Im folgenden wollen wir jedoch Turingmaschinen als Erkennungsgeräte und weniger als Rechengeräte auffassen. Dies stellt aber keine Einschränkung dar. Ist z.B. bei Vorgabe eines Arguments a ein Funktionswert $f(a)$ zu berechnen, so kann dies auch als das Problem aufgefaßt werden zu erkennen, ob bei Vorgabe von zwei Argumenten diese a und $f(a)$ darstellen.

3.6 Modifizierte Turingmaschinen

Es gibt eine Reihe anderer Definitionen von Turingmaschinen, die an ihren prinzipiellen Möglichkeiten nichts verändern, wohl aber zu anderen Komplexitäten, z.B. bezüglich des Zeitbedarfs, führen können. Wir geben hier eine Aufstellung einiger modifizierter Turingmaschinen an. Sie werden anschaulich charakterisiert.

(a) Eine *Turingmaschine mit einseitig begrenztem Band* hat ein Band der Form

0	1	2	3	4	5	6	#

Sie hält, wenn sie sich über den linken Rand hinaus bewegt. Sie kann durch eine Turingmaschine mit beidseitig potentiell unendlichem Band

#	¢	0	1	2	3	4	5	6	#

simuliert werden, wobei die simulierende Turingmaschine hält, wenn sie links auf das Zeichen „¢“ trifft. Umgekehrt kann eine Turingmaschine gemäß Definition 3.13 durch eine Turingmaschine mit einseitig begrenztem Band simuliert werden. Die Zuordnung der Bänder erfolgt durch

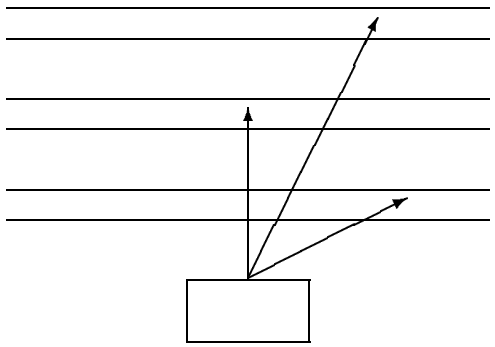
#	−5	−4	−3	−2	−1	0	1	2	3	4	5	#

⇓

0	1	2	3	4	5	#
¢	−1	−2	−3	−4	−5	

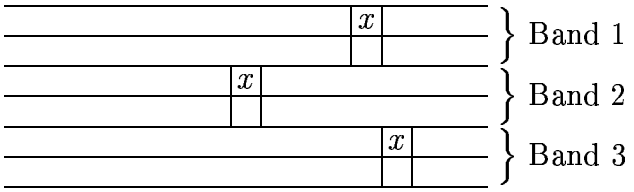
Diese Turingmaschine mit einseitig begrenztem Band besitzt ein Band, bei dem ein Feld zwei Einträge hat. Jedes Feld kann dann durch ein Element eines kartesischen Produkts von zwei Mengen dargestellt werden. Die Turingmaschine hat also ein Band mit zwei Spuren. Durch eine entsprechende Zustandskomponente kann man sich merken, ob gerade die untere oder obere Spur bearbeitet wird.

(b) Bei einer *Turingmaschine mit mehreren Bändern* bewegen sich alle Köpfe unabhängig voneinander, jedoch aufgrund der Gesamtinformation. Eine solche Turingmaschine mit drei Bändern kann durch das folgende Bild dargestellt werden.



Man kann z.B. annehmen, daß die Turingmaschine insgesamt stoppt, wenn sie auf mindestens einem Band hält. Eine solche Turingmaschine wird durch eine

Turingmaschine der Darstellung



simuliert, wobei die jeweils obere Spur der drei Bänder die Kopfstellung durch ein Zeichen x markiert und die jeweils untere Spur zur Bearbeitung des ursprünglichen Bandes dient. Ein Zustand dieser Turingmaschine besteht aus einem Zustand der simulierten Turingmaschine, einem Kopfzähler und zusätzlichen Informationen. Die Maschine beginnt beim linken Kopf mit 1 als Wert des Kopfzählers. Sie läuft dann nach rechts, wobei sie das jeweilige Symbol unter x in den Zustand aufnimmt und den Kopfzähler um 1 erhöht. Beim rechten Kopf, der durch den Wert 3 des Kopfzählers identifizierbar ist, sind alle nötigen Informationen zur Simulation der oberen Turingmaschine gesammelt. Die Maschine läuft nun nach links, wobei die Operationen des jeweiligen Kopfes der oberen Turingmaschine ausgeführt werden. Beim linken Kopf erfolgt schließlich zusätzlich eine Änderung des Zustandes gemäß der simulierten Turingmaschine sowie ein Zurücksetzen des Kopfzählers auf 1. Umgekehrt kann eine Turingmaschine gemäß Definition 3.13 als eine Turingmaschine mit n Bändern aufgefaßt werden. Auf einem Band erfolgt die Arbeit wie auf der gegebenen Maschine. Die anderen Bänder und Köpfe werden nicht verändert bzw. bewegt.

(c) Eine *Turingmaschine mit mehreren Köpfen auf einem Band* wird ähnlich wie die Turingmaschine aus (d) simuliert.

(d) Bei einer *Turingmaschine mit zweidimensionalem Band* erfolgt die Bewegung des Kopfes nach links, rechts, oben oder unten. Die folgenden Überlegungen können auch auf Maschinen mit n-dimensionalem Band übertragen werden. Wir betrachten das zweidimensionale Band

$$
\begin{array}{ccccccc}
 & & & \# & & & \\
 & B & B & x_1 & B & x_2 & \\
\# & x_3 & B & B & x_4 & B & \# \\
 & B & x_5 & x_6 & B & x_7 & \\
 & & & \# & & &
\end{array}
$$

Es wird durch ein Band $\#*BBx_1Bx_2*x_3BBx_4B*Bx_5x_6Bx_7*\#$ einer Turingmaschine mit zwei Bändern kodiert, wobei auf dem zweiten Band vermerkt werden kann, wieviele Symbole in einer Zeile stehen. Die Bewegungen der zweidimensionalen Turingmaschine können durch solche Turingmaschinen mit zwei Bändern simuliert werden. Wird dabei z.B. außerhalb des umrandeten Teils des Bandes ein neues Zeichen x geschrieben, also eine Bandvergrößerung vorgenommen, so müssen in der Kodierung außer x auch Blankzeichen eingefügt werden.

Zum Abschluß soll noch kurz auf die Existenz einer *universellen Turingmaschine* hingewiesen werden. Falls eine solche Maschine mit der Kodierung einer beliebigen Turingmaschine T sowie einer Eingabe w für T startet, so simuliert sie das Verhalten von T auf der Eingabe w. Sie hält also, wenn auch T hält. Handelt es sich um eine Turingmaschine T, die eine Funktion berechnet, so berechnet auch die universelle Turingmaschine den entsprechenden Funktionswert.

Kapitel 4

Charakterisierung regulärer Sprachen

4.1 Reguläre Ausdrücke

Wie wir auf Seite 19 bemerkt haben, besteht die Möglichkeit, eine Sprache durch ein Konstruktionsverfahren zu beschreiben. Dies soll hier für reguläre Sprachen durchgeführt werden.

Definition 4.1 Es seien V und $V' = \{\cup, ^*, \emptyset, (,)\}$ disjunkte Alphabete. Es heißt $P \in (V \cup V')^*$ *regulärer Ausdruck über* V, wenn die folgenden Aussagen erfüllt sind:

(a) $P \in V \ \vee \ P = \emptyset$ oder

(b) $P = (Q \cup R) \ \vee \ P = (QR) \ \vee \ P = Q^*$
mit regulären Ausdrücken Q und R über V. $\square$

Definition 4.2 Es sei P ein regulärer Ausdruck über V. Dann *bezeichnet* P *die Sprache* $< P >$ über V, wobei $< P >$ wie folgt gegeben ist:

(a) Die durch $\emptyset$ bezeichnete Sprache ist die leere Sprache.

(b) Die durch $a \in V$ bezeichnete Sprache besteht aus dem Wort a, d.h. $< a >= \{a\}$.

(c) Für reguläre Ausdrücke P und Q über V gilt

$$< (P \cup Q) >=< P > \cup < Q >, \ < (PQ) >=< P >< Q >, \ < P^* >=< P >^* .$$

$\square$

Man beachte, daß formal das Zeichen $\cup$ in einem regulären Ausdruck nicht das Vereinigungszeichen zweier Mengen ist. In $< (P \cup Q) >=< P > \cup < Q >$ bedeutet z.B. nur das zweite $\cup$-Zeichen die Mengenvereinigung, entsprechendes gilt für das *-Zeichen und auch die „Konkatenation“. Der Zusammenhang zwischen einem regulären Ausdruck und der durch ihn bezeichneten Sprache kann so gesehen werden, daß der reguläre Ausdruck der *syntaktische* Begriff ist, dessen *Semantik* durch die bezeichnete Sprache gegeben wird.

Wir vereinbaren, daß bei regulären Ausdrücken „unnötige“ Klammern weggelassen werden können. Die Stärke der Bindung ist durch die Reihenfolge „Iteration“, „Konkatenation“, „Vereinigung“ gegeben.

Beispiel 4.1 Wir wählen $V = \{a\}$ und erhalten damit z.B.

$$\begin{aligned} < (aa)^* > &= < (aa)^* \cup (\emptyset a) > = < (aa \cup aaaa)^* > = < (aa)^*(aa)^* > \\ &= \{w \mid w = a^{2n}, n \in \mathbb{N}_0\}, \\ < \emptyset^* > &= < \emptyset >^* = \emptyset^* = \{\varepsilon\}. \quad \square \end{aligned}$$

Satz 4.1 Es sei L eine Sprache. L ist genau dann durch einen regulären Ausdruck bezeichnet, wenn $L \in \mathfrak{L}_3$ gilt.

Beweis: Wir gehen zunächst davon aus, daß L durch einen regulären Ausdruck über $\{a_1, \ldots, a_r\}$ bezeichnet ist, d.h., L ergibt sich aus den endlichen Sprachen $\emptyset$, $\{a_1\}, \ldots, \{a_r\}$ durch endlich viele Anwendungen der regulären Operationen Vereinigung, Konkatenation und Iteration. Nach Satz 2.3 gilt dann $L \in \mathfrak{L}_3$.

Umgekehrt sei L eine Typ-3-Sprache. Dann wird L nach Satz 3.3 von einem deterministischen endlichen erkennenden Automaten E akzeptiert. Es sei V_T sein Terminalalphabet, $\{s_1, \ldots, s_n\}$ mit einem geeigneten $n \in \mathbb{N}$ seine Zustandsmenge, s_1 sei dabei der Anfangszustand, und die Endzustandsmenge werde durch $S_1 = \{s'_1, \ldots, s'_k\}$ gegeben, wobei $k \in \mathbb{N}_0$ mit $k \leq n$ gilt. Weiter sei F die Produktionenmenge von E. Dann definieren wir für $i = 1, \ldots, k$ endliche erkennende Automaten $E^{s'_i}$ dadurch, daß sie im wesentlichen wie E definiert sind, nur die Endzustandsmenge S_1 von E wird durch $\{s'_i\}$ ersetzt. Offenbar gilt

$$L(E) = \bigcup_{i=1}^{k} L(E^{s'_i}).$$

Ist nun $k = 0$, also $S_1 = \emptyset$, so wird $L(E)$ nach Definition 4.2(a) durch den regulären Ausdruck $\emptyset$ bezeichnet. Für $k > 0$ müssen wir nach Definition 4.2(c) zeigen, daß jede Sprache $L(E^{s'_i})$ durch einen regulären Ausdruck bezeichnet ist. Wir beweisen allgemein, daß dies für jede Sprache $L(E^{s_j})$, $j = 1, \ldots, n$, gilt.

Wir definieren

$$L_{ij}^k, \; i, j, k \in \mathbb{N}_0, \; 0 \leq k \leq n, \; 1 \leq i, j \leq n,$$

als die Sprache, die aus allen Wörtern w besteht, für die eine Ableitung

$$s_i w \Longrightarrow^* s_j \text{ in 0 oder einem Schritt}$$

möglich ist oder aber für die eine Ableitung

$$s_i w \Longrightarrow s_{i_1} w_1 \Longrightarrow \ldots \Longrightarrow s_{i_t} w_t \Longrightarrow s_j$$

existiert mit $i_\nu \leq k$, $\nu = 1, \ldots, t$ für ein $t \in \mathbb{N}$. Anschaulich ausgedrückt, ist L_{ij}^k die Menge der Wörter w, die den endlichen erkennenden Automaten E vom Zustand s_i in

den Zustand s_j überführen und dabei nur Zustände $s_{k'}$ mit $k' \leq k$ durchlaufen. Man beachte, daß $i, j > k$ gelten darf. Wir zeigen durch Induktion über k, daß L_{ij}^k durch einen regulären Ausdruck bezeichnet werden kann.

Nach Definition von L_{ij}^k gilt $\varepsilon \in L_{ij}^k$ genau dann, wenn $i = j$ ist. Wir betrachten L_{ij}^0. Für $i \neq j$ besteht L_{ij}^0 aus den Wörtern, die in einem Schritt den Zustand s_i in s_j überführen, so daß $L_{ij}^0 \subset V_T$ gilt. Für $i = j$ gilt noch zusätzlich $\varepsilon \in L_{ij}^k$ und damit $\varepsilon \in L_{ij}^k \subset V_T \cup \{\varepsilon\}$. Da nach Definition 4.2(a) und (c) die Sprache $\{\varepsilon\}$ durch den regulären Ausdruck $\emptyset^*$ und jede Teilmenge $\{a_1, \ldots, a_l\} \subset V_T$ nach Definition 4.2(b) und (c) durch den regulären Ausdruck $(a_1 \cup \ldots \cup a_l)$ bezeichnet ist, ist auch L_{ij}^0 für alle i, j mit $1 \leq i, j \leq n$ durch einen regulären Ausdruck bezeichnet.

Wir nehmen weiter an, daß für ein festes k, $0 \leq k \leq n-1$, jede Sprache L_{ij}^k durch einen regulären Ausdruck bezeichnet ist. Dann gilt offenbar

$$L_{ij}^{k+1} = L_{ij}^k \cup L_{i(k+1)}^k (L_{(k+1)(k+1)}^k)^* L_{(k+1)j}^k.$$

Nach Definition 4.2(c) ist dann auch L_{ij}^{k+1} durch einen regulären Ausdruck bezeichnet. Damit ist der Induktionsbeweis beendet.

Speziell erkennen wir, daß $L(E^{s_j}) = L_{1j}^n$ für alle $j = 1, \ldots, n$ durch einen regulären Ausdruck bezeichnet ist. □

Definition 4.3 Es sei $w = a_1 a_2 \ldots a_n$, $a_i \in V_T$, $i = 1, \ldots, n$, $n \in \mathbb{N}_0$. Durch $sp(w) = a_n \ldots a_2 a_1$ wird die *Spiegeloperation* $sp : V_T^* \to V_T^*$ definiert. Speziell gilt $sp(\varepsilon) = \varepsilon$. Damit wird $sp(L) = \{w' \mid w' = sp(w), w \in L\}$ für jede Sprache $L \subset V_T^*$ gegeben.

Satz 4.2 Die Familie $\mathfrak{L}_3$ ist abgeschlossen unter Substitution, beliebigem Homomorphismus und der *sp*-Operation.

Beweis: Es sei $L \subset V_T^*$, $L \in \mathfrak{L}_3$. Wegen Satz 4.1 wird L durch einen regulären Ausdruck P über V_T bezeichnet. Wir beweisen zunächst den Abschluß unter der *sp*-Operation. P' sei der reguläre Ausdruck, der sich aus P durch Vertauschung der Faktoren in allen Konkatenationen ergibt. Dann ist offenbar $sp(L)$ durch P' bezeichnet, und es folgt $sp(L) \in \mathfrak{L}_3$. Weiter sei σ eine Substitution. Somit wird jedes $a \in V_T$ in den Wörtern von L durch $\sigma(a) \in \mathfrak{L}_3$ ersetzt. $\sigma(a)$ ist jedoch durch einen regulären Ausdruck Q_a bezeichnet. Wird jedes $a \in V_T$ in P durch Q_a ersetzt, so erhalten wir nach Definition 4.1 einen regulären Ausdruck P'', der die Sprache $\sigma(L)$ bezeichnet, d.h. $\sigma(L) \in \mathfrak{L}_3$. Ein Homomorphismus ist eine spezielle Substitution. □

4.2 Lineare Grammatiken

Satz 4.3 $\mathfrak{L}_3$ ist gleich der Familie der Sprachen, die von Grammatiken erzeugt werden, deren Produktionen von einer der folgenden Formen sind:

$$(*) \qquad X \to wY \text{ oder } X \to w \text{ mit } X, Y \in V_N, w \in V_T^*.$$

Beweis: Für jede Grammatik G sei $sp(G)$ die Grammatik, die sich aus G durch Ersetzung der rechten Seiten ihrer Produktionen durch das jeweilige Spiegelbild ergibt.

Es gelte $L \in \mathfrak{L}_3$. Nach Satz 4.2 folgt $sp(L) \in \mathfrak{L}_3$. Es sei G' eine Typ-3-Grammatik mit $L(G') = sp(L)$. Dann ist $sp(G')$ eine Grammatik, deren Produktionen die Form $(*)$ haben. Offenbar gilt $L(sp(G')) = sp(L(G')) = sp(sp(L)) = L$. Somit wird L von einer Grammatik mit Produktionen der Form $(*)$ erzeugt.

Umgekehrt werde L von einer Grammatik G' mit Produktionen der Form $(*)$ erzeugt. Dann ist $sp(G')$ eine Typ-3-Grammatik mit $L(sp(G')) = sp(L) = sp(L(G'))$. Es folgt $sp(L(G')) \in \mathfrak{L}_3$. Nach Satz 4.2 erhalten wir $L = L(G') = sp(sp(L(G'))) \in \mathfrak{L}_3$. □

Definition 4.4 Es sei $G = (V_N, V_T, X_0, F)$ eine Grammatik.

(a) G heißt *lineare Grammatik*, wenn die Produktionen von G von einer der beiden folgenden Formen sind:

$$X \to w_1 Y w_2, \; X \to w \text{ mit } X, Y \in V_N, \; w_1, w_2, w \in V_T^*.$$

(b) G heißt *rechts-lineare Grammatik*, wenn G linear ist und $w_2 = \varepsilon$ für alle w_2 aus (a) gilt.

(c) G heißt *links-lineare Grammatik*, wenn G linear ist und $w_1 = \varepsilon$ für alle w_1 aus (a) gilt.

(d) G heißt *einseitig linear*, wenn G entweder rechts- oder links-linear ist. □

Satz 4.4 Es existieren Sprachen, die von linearen Grammatiken, nicht aber von einseitig linearen Grammatiken erzeugt werden.

Beweis: Im Beweis von Satz 3.8 haben wir gezeigt, daß die Sprache $L_1 = \{a^n b a^n \mid n \in \mathbb{N}\}$ keine Typ-3-Sprache ist, also von keiner einseitig linearen Grammatik erzeugt wird. Andererseits wird sie jedoch von der linearen Grammatik $G = (\{X\}, \{a, b\}, X, \{X \to aXa, X \to aba\})$ erzeugt. □

4.3 Die Äquivalenzrelation von Nerode

Definition 4.5 Es sei V ein Alphabet, und es gelte $L \subset V^*$. Die Relation E_L auf V^* wird für alle $w_1, w_2 \in V^*$ durch

$$w_1 E_L w_2 \iff \bigwedge_{u \in V^*} (w_1 u \in L \iff w_2 u \in L)$$

gegeben. Sie heißt die *durch L induzierte Äquivalenzrelation auf V^** oder die *Äquivalenzrelation von Nerode.* □

Daß die Eigenschaften einer Äquivalenzrelation erfüllt sind, ist offensichtlich. Wir stellen weiter sofort fest, daß die Äquivalenzrelation von Nerode *rechtsinvariant* ist, d.h., es gilt:

$$\bigwedge_{u \in V^*} (w_1 E_L w_2 \Longrightarrow w_1 u E_L w_2 u).$$

Definition 4.6 Es sei E_L die Äquivalenzrelation von Nerode für eine Sprache $L \subset V^*$. Mit $\text{ind}(E_L)$ bezeichnen wir die Anzahl der Äquivalenzklassen von E_L, und wir nennen diese Zahl den *Index von* E_L. □

Satz 4.5 Eine Sprache L ist genau dann vom Typ 3, wenn $\text{ind}(E_L) < \infty$ gilt.

Beweis: Wir benutzen in diesem Beweis, daß nach Satz 3.3 eine Sprache genau dann vom Typ 3 ist, wenn sie von einem deterministischen endlichen erkennenden Automaten akzeptiert wird.

Es gelte $L \subset V_T^*$ mit $\text{ind}(E_L) = n + 1$ für ein $n \in \mathbb{N}_0$. Die Repräsentanten der Äquivalenzklassen seien $w_0 = \varepsilon$, $w_1, \ldots, w_n$, und $[w]$ sei die durch ein Wort w festgelegte Äquivalenzklasse. Wir konstruieren einen endlichen erkennenden Automaten $E = (S, V_T, \delta, s_0, S_1)$ (in der Schreibweise von Seite 37) mit $S = \{[w_0], [w_1], \ldots, [w_n]\}$, $s_0 = [w_0]$, $S_1 = \{[w_i] \mid w_i \in L,\ i = 0, \ldots, n\}$ sowie der Überführungsfunktion δ mit $\delta([w_i], a) = [w_i a]$ für alle $a \in V_T$ und $i = 0, \ldots, n$. Wenn wir $u = \varepsilon$ in Definition 4.5 einsetzen, erkennen wir, daß entweder alle Wörter einer Klasse in L liegen oder keins. Folglich kommt es bei der Definition von S_1 auf den speziellen Repräsentanten w_i nicht an. Da E_L rechtsinvariant ist, ist die Definition von δ nicht von der Wahl des Repräsentanten w_i abhängig, δ ist also wohldefiniert. Für beliebige $v \in V_T^*$ gilt offenbar

$$[w_0]v \Longrightarrow_E^* [v].$$

Nach Definition von S_1 werden genau die Wörter $v \in L$ akzeptiert. Folglich gilt $L = L(E)$.

Umgekehrt gelte $L = L(E)$ mit einem endlichen erkennenden Automaten $E = (S, V_T, \delta, s_0, S_1)$. Wir definieren eine Äquivalenzrelation $\tilde{E}$ auf V_T^* wie folgt:

$$w_1 \tilde{E} w_2 : \Longleftrightarrow \ (1) \quad \bigvee_{s \in S} (s_0 w_1 \Longrightarrow_E^* s \ \wedge \ s_0 w_2 \Longrightarrow_E^* s).$$

Die Eigenschaften einer Äquivalenzrelation sind offensichtlich erfüllt. Wir behaupten, daß die Klassen von $\tilde{E}$ feiner unterteilt sind als die von E_L, d.h., es soll

$$\bigwedge_{w_1, w_2 \in V_T^*} (w_1 \tilde{E} w_2 \Longrightarrow w_1 E_L w_2)$$

gelten. Wir nehmen das Gegenteil an, also

$$\bigvee_{w_1, w_2 \in V_T *} \neg(w_1 \tilde{E} w_2 \Longrightarrow w_1 E_L w_2) \Longleftrightarrow \bigvee_{w_1, w_2 \in V_T^*} (w_1 \tilde{E} w_2 \ \wedge \ w_1 \not{E}_L w_2)$$
$$\Longleftrightarrow \bigvee_{w_1, w_2, u \in V_T^*} (w_1 \tilde{E} w_2, \text{ genau eins der Wörter } w_1 u,\ w_2 u \text{ gehört zu } L).$$

Ohne Beschränkung der Allgemeinheit folgt

$$(2) \quad s_0 w_1 u \Longrightarrow_E^* s_1,\ s_1 \in S_1,\ s_0 w_2 u \Longrightarrow_E^* s_2,\ s_2 \notin S_1.$$

Wegen $w_1\tilde{E}w_2$ müssen (1) und (2) gleichzeitig erfüllt sein, was jedoch, da E ein deterministischer Automat ist, ein Widerspruch ist. Folglich gilt die Behauptung und damit

$$\text{ind}(E_L) \leq \text{ind}(\tilde{E}) \leq \text{card}(S) < \infty. \quad \square$$

Der Satz ist gut anwendbar, um zu überprüfen, ob eine vorgegebene Sprache zur Familie $\mathfrak{L}_3$ gehört oder nicht. Wir geben dazu ein Beispiel an.

Beispiel 4.2 Es sei $L = \{a^n b^n \mid n \in \mathbb{N}\}$. Wir nehmen an, daß L vom Typ 3 ist. Nach Satz 4.5 gilt dann $\text{ind}(E_L) < \infty$. Folglich müssen $m, n \in \mathbb{N}$ existieren mit $m > n$ und $a^m E_L a^n$. Nach Definition 4.5 erhalten wir

$$(a^m b^m \in L \iff a^n b^m \in L),$$

einen Widerspruch. Wir bemerken, daß diese Sprache jedoch kontextfrei und sogar linear ist, da sie von der Grammatik

$$G = (\{X_0\}, \{a, b\}, X_0, \{X_0 \to aX_0b, X_0 \to ab\})$$

erzeugt wird. Es handelt sich also auch hier um eine Sprache, die zum Beweis von Satz 4.4 herangezogen werden kann. $\square$

4.4 Typ-3-Sprachen und stochastische Akzeptoren

Wir kommen noch einmal auf stochastische Akzeptoren (siehe Definition 1.17) zurück. Nach Satz 1.10 läßt sich jeder endliche erkennende Automat als stochastischer Akzeptor auffassen. Satz 1.12 zeigte, daß die Familie der durch stochastische Akzeptoren erkannten Sprachen die Familie der regulären Sprachen echt umfaßt. Wir wollen angeben, welche anderen Einschränkungen gemacht werden können, um ebenfalls nur reguläre Sprachen zu erhalten.

Zunächst soll die folgende Definition motiviert werden. Es sei W ein stochastischer Akzeptor und λ sein Schnittpunkt. Die zugehörigen stochastischen Matrizen seien nicht explizit bekannt. Durch mehrmalige Eingabe eines Wortes w in den Akzeptor W soll getestet werden, ob $w \in L(W)$ gilt, d.h., ob $\pi \cdot \mathfrak{M}(w) \cdot \bar{\sigma}_1 > \lambda$ erfüllt ist. Dies kann etwa durch Aufleuchten einer Lampe festgestellt werden. Ist nun N die Gesamtanzahl der Versuche und m die Anzahl der Versuche, bei denen w von W akzeptiert wird, so gilt nach dem starken Gesetz der großen Zahlen

$$\lim_{N \to \infty} \left(\frac{m}{N}\right) = \pi \cdot \mathfrak{M}(w) \cdot \bar{\sigma}_1 = Pr(w).$$

Nach endlich vielen Versuchen wird das Verfahren abgebrochen. Die Entscheidung, ob $Pr(w) > \lambda$ gilt, ist nicht sicher. Falls nun λ ein Häufungspunkt von $\{\pi \cdot \mathfrak{M}(w) \cdot \bar{\sigma}_1 \mid w \in V_T^*\}$ ist, benötigt man im allgemeinen sehr viele Versuche, um mit einer großen Wahrscheinlichkeit angeben zu können, ob $w \in L(W)$ gilt oder nicht. Falls λ kein Häufungspunkt ist, kann die richtige Entscheidung mit großer Wahrscheinlichkeit schon mit weniger Versuchen gefällt werden. Solche Schnittpunkte heißen isoliert.

Definition 4.7 Es sei $W = (A, \lambda)$ ein stochastischer Akzeptor. λ heißt *isolierter Schnittpunkt* von W, wenn ein $\Delta > 0$ existiert, so daß für alle $w \in V_T^*$ die Beziehung $|Pr(w) - \lambda| \geq \Delta$ gilt. □

Die Werte von $Pr(w)$ liegen also niemals in der Δ-Umgebung von λ. Der etwas umfangreiche Beweis des folgenden Satzes soll hier nicht durchgeführt werden. Er kann z.B. in dem Buch von *Claus* [4] nachgelesen werden.

Satz 4.6 Es sei $W = (A, \lambda)$ ein stochastischer Akzeptor mit isoliertem Schnittpunkt λ. Dann ist $L(W)$ eine Typ-3-Sprache. Ist Δ die Schranke von λ und besitzt der stochastische Akzeptor n Zustände, so existiert ein endlicher erkennender Automat E mit $L(W) = L(E)$, der höchstens $1 + (1 + \frac{1}{2\Delta})^{n-1}$ Zustände besitzt. □

Wir bemerken, daß ein endlicher erkennender Automat als ein spezieller stochastischer Akzeptor mit isoliertem Schnittpunkt aufgefaßt werden kann.

4.5 Selbsteinbettende Typ-2-Grammatiken

Definition 4.8 (a) Es sei $G = (V_N, V_T, X_0, F)$ eine Typ-2-Grammatik. G heißt *selbsteinbettend*, wenn es ein $X \in V_N$ gibt sowie $P, Q \in V^+$ mit $X \Longrightarrow^* PXQ$.

(b) Es sei $L \in \mathfrak{L}_2$. L heißt *selbsteinbettend*, wenn für alle Typ-2-Grammatiken G mit $L(G) = L$ folgt, daß G selbsteinbettend ist. □

Satz 4.7 Es gilt $L \in \mathfrak{L}_3$ genau dann, wenn $L \in \mathfrak{L}_2$ gilt und L nicht selbsteinbettend ist.

Beweis: Es sei $L \in \mathfrak{L}_3$. Dann existiert eine Typ-3-Grammatik G mit $L(G) = L$. Als Typ-3-Grammatik ist G auch vom Typ 2, jedoch nicht selbsteinbettend. Es folgt $L \in \mathfrak{L}_2$, nach Definition 4.8 ist L jedoch nicht selbsteinbettend.

Umgekehrt werde L von einer Typ-2-Grammatik $G = (V_N, V_T, X_0, F)$ erzeugt, die nicht selbsteinbettend ist. Ohne Beschränkung der Allgemeinheit nehmen wir an, daß

$$X_0 \Longrightarrow^* S,\ S \in V^*\{X\}V^*,\ \text{für alle } X \in V_N$$

gilt. Anderenfalls ist X überflüssig und kann zusammen mit Produktionen, die X enthalten, entfernt werden, ohne die erzeugte Sprache zu ändern. Wir betrachten zwei Fälle.

Fall 1: Für alle $X \in V_N$ gilt $X \Longrightarrow^* S$, $S \in V^*\{X_0\}V^*$.

Jede Produktion von F mit Nichtterminalzeichen auf der rechten Seite ist von einer der folgenden Formen:

$$\begin{array}{llll} (1)\ X \to PYQ, & (2)\ X \to PY, & & \\ (3)\ X \to YQ, & (4)\ X \to Y & \text{für } X, Y \in V_N,\ P, Q \in V^+. & \end{array}$$

Wenn F eine Produktion der Form (1) enthält, dann gilt

$$X \Longrightarrow PYQ \Longrightarrow^* PP_1X_0Q_1Q \Longrightarrow^* PP_1P_2XQ_2Q_1Q$$

mit $PP_1P_2, Q_2Q_1Q \in V^+$. Nach Definition 4.8(a) ist somit G selbsteinbettend, ein Widerspruch. Wir nehmen nun an, daß F eine Produktion $X \to PY$ der Form (2) und gleichzeitig eine Produktion $X_1 \to Y_1Q$ der Form (3) enthält. Dann gilt

$$\begin{aligned} X \Longrightarrow PY \Longrightarrow^* PP_1X_0Q_1 \Longrightarrow^* PP_1P_2X_1Q_2Q_1 \Longrightarrow PP_1P_2Y_1QQ_2Q_1 \\ \Longrightarrow^* PP_1P_2P_3X_0Q_3QQ_2Q_1 \Longrightarrow^* PP_1P_2P_3P_4XQ_4Q_3QQ_2Q_1 \end{aligned}$$

mit $PP_1P_2P_3P_4$, $Q_4Q_3QQ_2Q_1 \in V^+$. Auch in diesem Fall erhalten wir einen Widerspruch.

F kann also höchstens Produktionen der Form (2) und (4), aber nicht der Form (1) oder (3), oder aber Produktionen der Form (3) und (4), aber nicht der Form (1) oder (2), enthalten. Für eine Produktion der Form (2), die nicht von der Form (1) oder (3) ist, muß jedoch $P \in V_T^+$ gelten. Folglich ist G rechts-linear, da auch die Produktionen der Form (4) rechts-linear sind. Nach Satz 4.3 gilt also $L \in \mathfrak{L}_3$. Enthält F dagegen Produktionen der Form (3), aber nicht der Form (1) oder (2), so ist G entsprechend links-linear, und es folgt wieder $L \in \mathfrak{L}_3$.

Fall 2: Es existiert $X_1 \in V_N$, so daß für keine $P, Q \in V^*$ die Relation $X_1 \Longrightarrow^* PX_0Q$ gilt.

Wir beweisen hier allgemein die folgende Behauptung:

> Ist G eine nicht selbsteinbettende Typ-2-Grammatik, dann folgt $L(G) \in \mathfrak{L}_3$.

Der Beweis erfolgt durch Induktion über die Anzahl n der Nichtterminalzeichen von G. Die Behauptung ist für den Fall 1 nach dem vorhergehenden Beweisteil immer richtig, es ist jeweils nur der Fall 2 zu betrachten. Für $n = 1$ tritt der Fall 2 wegen $X_0 \Longrightarrow^* X_0$ nicht auf. Die Induktionsannahme laute, daß die Behauptung für k richtig ist. Es sei $\mathrm{card}(V_N) = k + 1$, und die Grammatik G erfülle die Eigenschaft von Fall 2 mit dem Symbol $X_1 \in V_N$. Dann gilt $X_1 \neq X_0$. Wir konstruieren zu G die Grammatik

$$G_1 = (V_N - \{X_0\}, V_T, X_1, F'),$$

wobei F' aus F durch Entfernung aller Produktionen entsteht, die X_0 enthalten. Aufgrund der Voraussetzung von Fall 2 sind gemäß G und G_1 von dem Zeichen X_1 ausgehend jeweils dieselben Ableitungen möglich. Weiter definieren wir die Grammatik

$$G_2 = (V_N - \{X_1\}, V_T \cup \{X_1\}, X_0, F''),$$

wobei F'' aus F durch Entfernung aller Produktionen entsteht, die auf der linken Seite X_1 enthalten. Man beachte, daß hier X_1 ein terminales Zeichen ist. Laut Induktionsannahme gilt $L(G_1), L(G_2) \in \mathfrak{L}_3$. Definieren wir eine reguläre Substitution σ von $(V_T \cup \{X_1\})^*$ in V_T^* durch

$$\sigma(a) = \{a\} \text{ für } a \in V_T \text{ und } \sigma(X_1) = L(G_1),$$

so gilt offenbar $\sigma(L(G_2)) = L(G)$. Wegen Satz 4.2 folgt $L(G) \in \mathfrak{L}_3$. $\square$

4.6 Zusammenfassung

Satz 4.8 Es sei L eine Sprache. Die folgenden Aussagen sind äquivalent:

(a) Es ist L eine Typ-3-Sprache.
(b) L wird von einer rechts-linearen Grammatik erzeugt.
(c) L wird durch einen regulären Ausdruck bezeichnet.
(d) L wird von einem endlichen erkennenden Automaten akzeptiert.
(e) L wird von einem nichtdeterministischen endlichen erkennenden Automaten akzeptiert.
(f) L ist eine Typ-2-Sprache und nicht selbsteinbettend.
(g) Die durch L induzierte Äquivalenzrelation E_L von Nerode ist von endlichem Index.
(h) L wird von einem stochastischen Akzeptor mit isoliertem Schnittpunkt erkannt. □

Kapitel 5

Eigenschaften kontextfreier Sprachen

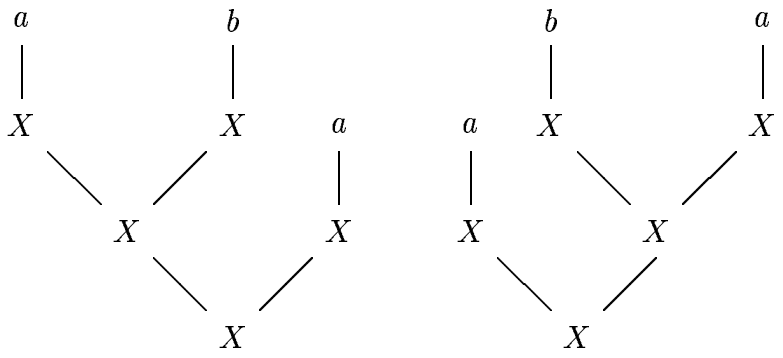
5.1 Ableitungsbäume und Mehrdeutigkeit von Ableitungen

Bei der Benutzung von kontextfreien Sprachen als Programmiersprachen ist es nicht nur von Bedeutung, ob ein vorgelegtes Wort, also ein Programm, zu der Sprache gehört, sondern für die Übersetzung ist auch die Art des Entstehens wichtig. Die möglichen Ableitungen werden in natürlicher Weise als *Ableitungsbäume* (bzw. Erzeugungsbäume, Zerlegungsbäume oder Zerlegungsdiagramme) charakterisiert.

Beispiel 5.1 Es sei $G = (\{X\}, \{a, b\}, X, \{X \to a, X \to b, X \to XX\})$ eine kontextfreie Grammatik. Es gilt offenbar $L(G) = \{a, b\}^+$. Zur Erzeugung des Wortes $aba \in L(G)$ gibt es unter anderem folgende Möglichkeiten:

$$\begin{array}{lccccccc}
(1) & X & \Longrightarrow & XX & \Longrightarrow & XXX & \Longrightarrow & aXX \Longrightarrow abX \Longrightarrow aba \\
(2) & X & \Longrightarrow & XX & \Longrightarrow & Xa & \Longrightarrow & XXa \Longrightarrow aXa \Longrightarrow aba \\
(3) & X & \Longrightarrow & XX & \Longrightarrow & aX & \Longrightarrow & aXX \Longrightarrow abX \Longrightarrow aba
\end{array}$$

Diese Ableitungen lassen sich durch zwei Ableitungsbäume darstellen:



Ableitungsbaum für (1) und (2)

Ableitungsbaum für (1) und (3) □

Das Konzept der Ableitungsbäume soll formalisiert werden.

Definition 5.1 Es sei T eine endliche, nichtleere Menge und $U \subset T \times T$. Die Elemente von T werden *Knoten*, die von U *Kanten* genannt. Man sagt, die Kante $(t_1, t_2) \in U$ *führt von* t_1 *in* t_2 (*geht von* t_1 *aus und mündet in* t_2). Eine endliche Folge von Kanten $e_1, \ldots, e_m$ heißt *Pfad* von der ersten Komponente von e_1 zur zweiten Komponente von e_m, wenn für $i = 1, \ldots, m-1$ die zweite Komponente von e_i mit der ersten Komponente von e_{i+1} übereinstimmt. Das Paar (T, U) (oder kurz T) heißt *Baum*, wenn die folgenden Eigenschaften erfüllt sind:

(a) Es existiert genau ein Knoten, *Wurzel* genannt, in den keine Kante führt.

(b) Für jeden Knoten t, der nicht die Wurzel ist, existiert genau ein Pfad von der Wurzel nach t. □

Wir sehen, daß nach der Definition ein Baum keine Schleifen enthält. Knoten, von denen keine Kanten wegführen, werden *Blätter* genannt.

Wir beschreiben nun ein Verfahren, wie man zu einer gegebenen kontextfreien Grammatik einen Ableitungsbaum konstruiert. Dabei werden die Knoten des Baumes mit terminalen oder nichtterminalen Zeichen oder dem leeren Wort markiert. Insbesondere werden die Knoten, die keine Blätter sind, durch nichtterminale Zeichen markiert. Wie diese Markierungen vorgenommen werden, wird im folgenden erläutert.

Wir gehen von einer kontextfreien Grammatik

$$G = (V_N, V_T, X_0, F)$$

sowie einer Ableitung

$$(1) \quad X = P_0 \Longrightarrow_G P_1 \Longrightarrow_G \ldots \Longrightarrow_G P_r$$

mit $X \in V_N$, $r \in \mathbb{N}$ aus. Für die dabei benutzten Produktionen $Y_i \to Q_i$, $i = 0, \ldots, r-1$, gelte

$$(2) \quad P_i = \alpha_i Y_i \beta_i, \; P_{i+1} = \alpha_i Q_i \beta_i$$

mit geeigneten Wörtern α_i und β_i. Speziell gilt also $Y_0 = X$ und $Q_0 = P_1$. Die Knoten des (1) beschreibenden Ableitungsbaums sind k-Tupel ($1 \leq k \leq r+1$) von natürlichen Zahlen, die ebenso wie ihre Markierungen wie folgt induktiv definiert sind:

(a) Es ist das 1-Tupel (1) die Wurzel des Baumes mit der Markierung X.

(b) Annahme: Für alle j, $0 \leq j \leq i$ mit $i < r$, sei jedes Zeichen in P_j bereits die Markierung irgendeines Knotens. Es sei $(u_1, \ldots, u_t)$ der Knoten, dessen Markierung das in (2) angegebene Y_i ist.

(c) (α) Für $Q_i = \varepsilon$ ist $(u_1, \ldots, u_t, 1)$ ein neuer Knoten mit der Markierung ε.

(β) Für $Q_i = x_1 \ldots x_m$, $x_u \in V_N \cup V_T$, $u = 1, \ldots, m$, erhalten wir für alle $u = 1, \ldots, m$ neue Knoten $(u_1, \ldots, u_t, u)$ mit der Markierung x_u.

Für je zwei Knoten $(u_1, \ldots, u_t)$, $(u_1, \ldots, u_t, u_{t+1})$ existiert eine Kante

$$((u_1, \ldots, u_t), (u_1, \ldots, u_t, u_{t+1})),$$

und alle Kanten sind von dieser Form.

Beispiel 5.2 Die Knoten des linken Baumes aus Beispiel 5.1 mit den zugeordneten Markierungen sind

(1)	(1,1)	(1,2)	(1,1,1)	(1,1,2)	(1,2,1)	(1,1,1,1)	(1,1,2,1)
X	X	X	X	X	a	a	b

□

Wenn wir im folgenden Ableitungsbäume benutzen, so werden wir sie nicht in dieser formalen Weise definieren, sondern vielmehr, wie schon in Beispiel 5.1 geschehen, ein anschauliches Bild angeben.

Wir schränken in der folgenden Definition den Begriff Ableitung so ein, daß jeder Baum der Ableitungsbaum von genau einer Ableitung ist.

Definition 5.2 Es sei $G = (V_N, V_T, X_0, F)$ eine kontextfreie Grammatik und

$$X = P_0 \Longrightarrow P_1 \Longrightarrow \ldots \Longrightarrow P_r$$

mit $X \in V_N$, $r \in \mathbb{N}$ eine Ableitung in G. Diese Ableitung heißt *Linksableitung*, wenn es für alle $i = 1, \ldots, r-1$ eine Produktion $Y_i \to Q_i$ gibt, so daß $P_i = \alpha_i Y_i \beta_i$, $P_{i+1} = \alpha_i Q_i \beta_i$ und $\alpha_i \in V_T^*$ für geeignete α_i und β_i gilt. □

In einer Linksableitung wird das am weitesten links stehende Nichtterminalzeichen ersetzt. Wir schreiben auch „$\Longrightarrow_{\text{links}}$“ für einen Linksableitungsschritt. Offenbar besteht eine bijektive Beziehung zwischen Linksableitungen und Ableitungsbäumen.

Analog zu Definition 5.2 können auch *Rechtsableitungen* definiert werden.

Definition 5.3 (a) Es sei G eine kontextfreie Grammatik. G heißt *mehrdeutig*, wenn ein Wort $w \in L(G)$ existiert, für das es mindestens zwei verschiedene Linksableitungen vom Anfangssymbol X_0 aus gibt.
G heißt *eindeutig*, wenn G nicht mehrdeutig ist.

(b) Es sei L eine kontextfreie Sprache. L heißt *eindeutig*, wenn eine eindeutige kontextfreie Grammatik G mit $L(G) = L$ existiert.
L heißt (*echt* oder *inhärent*) *mehrdeutig*, wenn L nicht eindeutig ist. □

Beispiel 5.3 (a) Es sei G die Grammatik aus Beispiel 5.1. Sie ist mehrdeutig, da die Ableitungen (1) und (3) verschiedene Linksableitungen für das Wort *aba* darstellen.

(b) Die Sprache

$$\{a^n b^n c^m d^m \mid m, n \in \mathbb{N}\} \cup \{a^n b^m c^m d^n \mid m, n \in \mathbb{N}\}$$

ist kontextfrei und außerdem echt mehrdeutig. Ein Beweis dafür findet sich im Buch von *Hopcroft/Ullman* [15], S. 99–103. Die ähnliche Sprache

$$\{a^m b^n c^n \mid m, n \in \mathbb{N}\} \cup \{a^m b^m c^n \mid m, n \in \mathbb{N}\}$$

ist ebenfalls echt mehrdeutig. Ein Beweis kann in dem Buch von *Salomaa* [24], Satz VI.9.3, nachgelesen werden und wird auch in Kapitel 12 dieses Skriptes behandelt. □

Satz 5.1 Jede Typ-3-Sprache ist eindeutig.

Beweis: Es sei $L \subset V_T^*$ eine Typ-3-Sprache. Dann existiert nach Satz 4.8 ein endlicher erkennender Automat $E = (V, F)$ mit $L(E) = L$. Dabei gelte $V = S \cup V_T$, s_0 sei der Anfangszustand und S_1 die Menge der Endzustände. Dann folgt, daß L von der kontextfreien (speziell rechts-linearen) Grammatik

$$G = (S, V_T, s_0, \{(s_i, as_j) \mid (s_i a, s_j) \in F,\ s_i, s_j \in S, a \in V_T\} \cup \{(s_i, \varepsilon) \mid s_i \in S_1\})$$

erzeugt wird. G ist eindeutig, da bei deterministischen Automaten für alle s_i und a der Folgezustand s_j eindeutig bestimmt ist und es somit auch für s_i und a nur genau eine Produktion $s_i \to as_j$ gibt. Für ein Wort $w = a_1 \dots a_n \in L(G)$ muß im i'-ten Schritt, $i' = 1, \dots, n$, genau das Symbol $a_{i'}$ erzeugt werden. Die Folge der anzuwendenden Produktionen ist also eindeutig bestimmt. Somit gibt es nur eine Linksableitung für das Wort w. □

5.2 Kontextfreie Sprachen als kontextsensitive Sprachen

In diesem Abschnitt soll gezeigt werden, daß sich zu jeder kontextfreien Grammatik eine äquivalente kontextsensitive Grammatik konstruieren läßt.

Definition 5.4 Es sei G eine Grammatik. G heißt *ε-frei*, wenn für alle Produktionen $(P, Q) \in F$ von G die Relation $Q \neq \varepsilon$ gilt. Eine Sprache L heißt *ε-frei*, wenn $\varepsilon \notin L$ gilt □

Satz 5.2 (a) Es sei G eine kontextfreie Grammatik. Dann existiert eine ε-freie kontextfreie Grammatik G_1 mit $L(G_1) = L(G) - \{\varepsilon\}$.

(b) Es sei G eine kontextfreie Grammatik mit $\varepsilon \in L(G)$. Dann existiert eine kontextfreie Grammatik $G' = (V'_N, V_T, X'_0, F')$ mit $L(G') = L(G)$ und der Eigenschaft, daß $(X'_0, \varepsilon) \in F'$ die einzige Produktion von G' mit rechter Seite ε ist, und X'_0 kommt außerdem auf der rechten Seite keiner anderen Produktion aus G' vor.

Beweis: Wir beginnen mit dem Beweis von (a). Es sei $G = (V_N, V_T, X_0, F)$ eine kontextfreie Grammatik. Wir konstruieren eine kontextfreie Grammatik $G_1 = (V_N, V_T, X_0, F_1)$ wie folgt. Wir setzen

$$\begin{aligned} U_1 &= \{X \mid (X, \varepsilon) \in F\}, \\ U_{i+1} &= U_i \cup \{X \mid (X, P) \in F,\ P \in U_i^*\} \text{ für } i \geq 1. \end{aligned}$$

Offenbar ist U_i, $i \in \mathbb{N}$, eine monoton wachsende Folge von Teilmengen der endlichen Menge V_N. Diese muß konstant werden, d.h., es existiert ein $k \in \mathbb{N}$ mit

$$U_k = U_{k+1},\ \bigwedge_{\nu \in \mathbb{N}} U_k = U_{k+\nu},\ (X \Longrightarrow_G^* \varepsilon \iff X \in U_k).$$

Wir erhalten also

$$\varepsilon \in L(G) \iff X_0 \in U_k.$$

F_1 besteht nun genau aus den Produktionen (X, P_1) mit $P_1 \neq \varepsilon$, für die eine Produktion $(X, P) \in F$ existiert, so daß P_1 aus P durch Streichen von ≥ 0 vorkommenden Elementen aus U_k entsteht.

Wir stellen zunächst fest, daß $L(G_1) \subset L(G) - \{\varepsilon\}$ gilt. Produktionen aus G_1, die nicht bereits zu G gehören, sind nämlich aus Produktionen von G durch Streichen von $l \geq 1$ Nichtterminalzeichen $X \in U_k$ entstanden, also durch Streichen von Zeichen mit $X \Longrightarrow_G^* \varepsilon$. Diese Produktionen können durch mindestens $l+1$ Ableitungsschritte gemäß G simuliert werden, wobei in mindestens l Schritten die entsprechenden Nichtterminalzeichen gelöscht werden.

Umgekehrt gilt $L(G) - \{\varepsilon\} \subset L(G_1)$. Um dies zu beweisen, definieren wir die Grammatik $G_2 = (V_N, V_T, X_0, F \cup F_1)$. Offenbar ist $L(G) \subset L(G_2)$ erfüllt. Wir betrachten ein Wort $w \in L(G_2)$, $w \neq \varepsilon$, und zeigen, daß w gemäß G_1 abgeleitet werden kann. Da $F - \{(X, \varepsilon) \mid (X, \varepsilon) \in F\} \subset F_1$ gilt, reicht es zu zeigen, wie eine Produktion (X, ε) in der Ableitung von w entfernt werden kann. Man suche den Ableitungsschritt, der die Produktion benutzt, die X einführt. Diese Produktion ersetze man an dieser Stelle durch diejenige von F_1, die sich aus der gegebenen durch Streichen von X auf der rechten Seite ergibt. Dies Verfahren wird wiederholt, bis keine Produktionen $(X, \varepsilon) \in F$ mehr in den Ableitungsschritten benutzt werden. Es folgt $w \in L(G_1)$ und damit $L(G) - \{\varepsilon\} \subset L(G_2) - \{\varepsilon\} \subset L(G_1)$.

Der Beweis von (b) ist jetzt sehr einfach. Wir konstruieren G_1 wie in (a). Dann definieren wir

$$G' = (V_N \cup \{X_0'\}, V_T, X_0', F_1 \cup \{X_0' \to X_0, X_0' \to \varepsilon\}).$$

Es folgt $L(G') = L(G_1) \cup \{\varepsilon\} = L(G)$, und G' besitzt die geforderten Eigenschaften. □

Beispiel 5.4 Wir führen die Konstruktion zum Teil (a) des Satzes 5.2 an einem einfachen Beispiel durch. Es sei

$$G = (\{X_0, X, Y\}, \{a, b\}, X_0, \{X_0 \to XY, X \to aY, Y \to b, X \to \varepsilon, Y \to \varepsilon\}).$$

Dann gilt $U_1 = \{X, Y\}$, $U_2 = \{X_0, X, Y\}$. Damit ist die Produktionenmenge von G_1 durch

$$X_0 \to XY,\ X_0 \to X,\ X_0 \to Y,\ X \to aY,\ X \to a,\ Y \to b$$

gegeben. □

Als unmittelbare Folgerung des Satzes erhalten wir:

Satz 5.3 Jede kontextfreie Sprache ist auch kontextsensitiv. □

5.3 Normalformen für kontextfreie Grammatiken

Im folgenden wollen wir zwei Normalformen von kontextfreien Grammatiken angeben. Wir werden zeigen, daß zu jeder ε-freien kontextfreien Grammatik äquivalente Grammatiken in den Normalformen existieren. Diese Normalformen sind in vielen Beweisen nützlich, da sie in bezug auf beliebige kontextfreie Grammatiken eingeschränkte Eigenschaften besitzen.

Definition 5.5 Es sei $G = (V_N, V_T, X_0, F)$ eine kontextfreie Grammatik. G ist in *Chomskyscher Normalform*, wenn alle Produktionen von G eine der folgenden Formen haben:

$$X \to YZ \text{ oder } X \to a \text{ für } X, Y, Z \in V_N,\ a \in V_T. \quad \square$$

Wir erkennen sofort, daß das leere Wort ε nicht in einer von einer kontextfreien Grammatik in Chomskyscher Normalform erzeugten Sprache enthalten sein kann. Aufgrund von Satz 5.2 ist es aber immer möglich, zu einer beliebigen Grammatik G eine Grammatik G_1 in Chomskyscher Normalform mit $L(G) - \{\varepsilon\} = L(G_1)$ zu konstruieren. Mit der Konstruktion zum Teil (b) von Satz 5.2 kann dann gegebenenfalls das leere Wort hinzugefügt werden. In der Literatur wird deshalb gelegentlich (siehe z.B. [11], Seite 104) für eine Grammatik in Chomskyscher Normalform auch eine Produktion $X_0 \to \varepsilon$ zugelassen, wobei dann X_0 nicht auf der rechten Seite irgendeiner anderen Produktion vorkommen darf.

Satz 5.4 Es sei L eine ε-freie kontextfreie Sprache. Dann existiert eine Grammatik G' in Chomskyscher Normalform mit $L = L(G')$.

Beweis: Nach Satz 5.2 können wir annehmen, daß L von einer ε-freien kontextfreien Grammatik $G = (V_N, V_T, X_0, F)$ erzeugt wird. Zunächst wollen wir Produktionen der Form

$$X \to Y \text{ mit } X, Y \in V_N$$

entfernen. Für ein beliebiges $X \in V = V_N \cup V_T$ setzen wir

$$\begin{aligned} U_1(X) &= \{X\}, \\ U_{i+1}(X) &= U_i(X) \cup \{Y \in V \mid \bigvee_{Z \in U_i(X)} (Z, Y) \in F\} \text{ für } i \geq 1. \end{aligned}$$

In $U_{i+1}(X)$ liegen die Zeichen aus V, die man in höchstens i Schritten aus X ableiten kann, wobei jedes Wort der Ableitung aus jeweils einem Zeichen besteht. Da $\mathrm{card}(V)$ endlich ist, folgt die Existenz eines $k \in \mathbb{N}$ mit $U_k(X) = U_{k+1}(X)$. Dann erhalten wir

$$U_k(X) = U_{k+\nu}(X) \text{ für alle } \nu = 1, 2, \ldots .$$

Man beachte dabei, daß sich im allgemeinen für verschiedene X auch verschiedene k ergeben. Im folgenden sei k das Maximum dieser verschiedenen Werte. Da G ε-frei ist, erhalten wir für $X \in V_N$, $Y \in V$ die Äquivalenz

$$X \Longrightarrow_G^* Y \iff Y \in U_k(X).$$

Speziell für $a \in V_T$ ergibt sich

$$a \in L(G) \iff a \in U_k(X_0).$$

Wir konstruieren eine Grammatik $G_1 = (V_N, V_T, X_0, F')$ durch Angabe der Produktionenmenge F':

(1) $(X_0, a) \in F'$ für alle $a \in V_T \cap U_k(X_0)$.
(2) $(Y, y_1 y_2 \dots y_n) \in F'$ mit $Y \in V_N$, $y_j \in V$, $j = 1, \dots, n$, $n \geq 2$, für die $(X, x_1 x_2 \dots x_n) \in F$ existiert mit $X \in U_k(Y)$, $y_j \in U_k(x_j)$ für alle $j = 1, \dots, n$.

Offenbar enthält G_1 keine Produktionen (X, Y) mit $X, Y \in V_N$. Aus (1) folgt, daß G und G_1 dieselben Wörter der Länge 1 erzeugen. Jede Produktion (2) wird offenbar nach Definition gemäß G simuliert. Umgekehrt liegen alle Produktionen $(X, x_1 x_2 \dots x_n) \in F$ wie in (2) schon in F'. Anwendungen von Produktionen $(X', Y') \in F$ in G können in der Form

$$Y \to \dots \to X \to \begin{array}{l} x_1 \to \dots \to y_1 \\ \vdots \\ x_n \to \dots \to y_n \end{array}$$

vorkommen. Die Übergänge von x_ν nach y_ν in G können verschieden lang sein und erfolgen natürlich nicht parallel. Die Anwendungen dieser Produktionen können in G_1 durch die Produktion $(Y, y_1 \dots y_n) \in F'$ simuliert werden. Wir erhalten also $L(G) = L(G_1)$.

Im nächsten Schritt wird G_1 durch die Grammatik $G_2 = (V_N'', V_T, X_0, F'')$ ersetzt, bei der alle Produktionen, die Terminalsymbole enthalten, die Form (X, a) mit $X \in V_N''$, $a \in V_T$ besitzen. Dies wird erreicht, indem die Produktionen des Typs (2) mit Hilfe der Konstruktion von Satz 2.2 ersetzt werden. Wir erhalten so $L(G_1) = L(G_2)$. Dabei werden keine Produktionen (X, Y) mit $X, Y \in V_N''$ eingeführt. Produktionen von G_2, die keine Terminalsymbole enthalten, haben die Gestalt

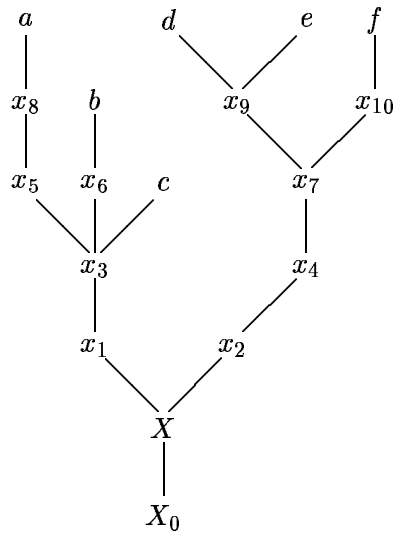
$$(X, Y_1 \dots Y_n) \text{ mit } n \geq 2, Y_j \in V_N'' \text{ für } j = 1, \dots, n.$$

Abschließend wird eine Grammatik G' in Chomskyscher Normalform konstruiert. Für jede Produktion der obigen Form mit $n > 2$ werden neue Nichtterminalzeichen $Z_1, \dots, Z_{n-2}$ eingeführt. Die obige Produktion wird dann durch die Produktionen

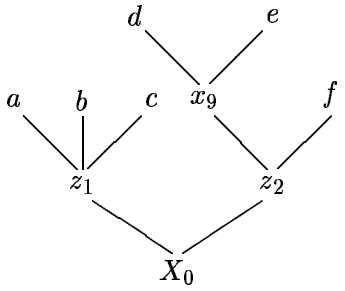
$$X \to Y_1 Z_1,\ Z_1 \to Y_2 Z_2,\ \dots, Z_{n-2} \to Y_{n-1} Y_n$$

ersetzt. Diese können nur in der angegebenen Reihenfolge angewendet werden. Es folgt $L(G') = L(G_2) = L(G_1) = L(G) = L$. □

Beispiel 5.5 Um den Übergang von der Grammatik G zur Grammatik G_1 im Beweis von Satz 5.4 zu verdeutlichen, betrachten wir einen Ableitungsbaum für das Wort $abcdef$ gemäß G, wobei die Produktionen implizit durch den Baum gegeben sind:



Ein Ableitungsbaum von $abcdef$ gemäß G_1 wird durch den folgenden Baum dargestellt, wobei die Markierung z_1 wahlweise für x_1 oder x_3 und die Markierung z_2 für x_2, x_4 oder x_7 steht.



□

Definition 5.6 Es sei $G = (V_N, V_T, X_0, F)$ eine kontextfreie Grammatik. G ist in *Greibachscher Normalform*, wenn alle Produktionen von G die Form

$$X \to aP \text{ mit } a \in V_T \text{ und } P \in V_N^*$$

besitzen. □

Es gelten auch hier entsprechende Bemerkungen wie nach Definition 5.5. Als Hilfssatz für Satz 5.6 benötigen wir

Satz 5.5 Es sei G eine ε-freie kontextfreie Grammatik. Dann existiert eine ε-freie kontextfreie Grammatik $\tilde{G} = (\tilde{V}_N, V_T, X_0, \tilde{F})$ mit $L(G) = L(\tilde{G})$ und der Eigenschaft, daß keine Produktion von $\tilde{G}$ von der Form

$$X \to Xv \text{ mit } X \in \tilde{V}_N,\ v \in (\tilde{V}_N \cup V_T)^*$$

ist.

Beweis: Die Aussage bedeutet, daß in $\tilde{G}$ keine links-rekursiven Produktionen vorhanden sind.

Wir nehmen an, daß es in G nur zwei Produktionen

$$X \to Xv \text{ und } X \to w \text{ mit } v, w \in (V_N \cup V_T)^+$$

gibt. Dabei beginne w nicht mit X. Dann ist aus X die Menge wv^* ableitbar, wobei, falls v oder w das Zeichen X enthält, daraus noch weitere Wörter ableitbar sind. Die Wörter aus wv^* erhält man äquivalent auch mit einem neuen Nichtterminalzeichen Z mit Hilfe der Produktionen

$$X \to w,\ X \to wZ, Z \to v,\ Z \to vZ.$$

Allgemein stellen wir zunächst fest, daß Produktionen $X \to X$ überflüssig und Produktionen $X \to Xv$ ohne eine Produktion $X \to w$ sinnlos sind. Solche Produktionen werden aus G entfernt. Dann betrachten wir für jedes $X \in V_N$ alle Produktionen

$$X \to Xv_1, \ldots, X \to Xv_n,\ X \to w_1, \ldots, X \to w_m,$$

wobei $v_i, w_j \in (V_N \cup V_T)^+$ gilt und w_j nicht mit X beginnt $(i = 1, \ldots, n,\ j = 1, \ldots, m)$. Diese werden ohne Änderung der erzeugten Sprache durch

$$\begin{aligned} &X \to w_1, \ldots, X \to w_m,\ X \to w_1Z, \ldots, X \to w_mZ, \\ &Z \to v_1, \ldots, Z \to v_n,\ Z \to v_1Z, \ldots, Z \to v_nZ \end{aligned}$$

ersetzt, wobei Z für jedes X jeweils ein neues Nichtterminalzeichen ist. Mit beiden Mengen von Produktionen ist aus X nämlich die Menge $\{w_1, \ldots, w_m\}\{v_1, \ldots, v_n\}^*$ erzeugbar. Damit ist $\tilde{G}$ konstruiert, und es gilt offenbar $L(\tilde{G}) = L(G)$. □

Wir bemerken, daß Z nicht in der ersten Position der rechten Seite irgendeiner Produktion von $\tilde{G}$ steht.

Satz 5.6 Es sei L eine ε-freie kontextfreie Sprache. Dann existiert eine Grammatik G' in Greibachscher Normalform mit $L = L(G')$.

Beweis: Wir gehen von einer ε-freien kontextfreien Grammatik $G = (V_N, V_T, X_0, F)$ mit $L(G) = L$ aus. Nach Satz 2.2 können wir ohne Beschränkung der Allgemeinheit annehmen, daß die einzigen Produktionen mit Terminalzeichen von der Form $A \to b$ mit $A \in V_N$, $b \in V_T$ sind. Die Elemente von V_N seien in einer festen Ordnung gegeben: $A_1, A_2, \ldots, A_n$. Eine Produktion

$$A_j \to w \text{ geht } \textit{bergauf}, \text{ wenn } w = A_kv \text{ mit } j < k \text{ oder } w = av \text{ mit } a \in V_T$$

gilt. Dabei muß für $w = av$ wegen unserer Annahme zunächst $v = \varepsilon$ gelten. Bei der folgenden Erweiterung von V_N zu einer größeren Menge von Nichtterminalzeichen wird v ein Nichtterminalwort sein.

Wir betrachten nun alle A_1-Produktionen, also Produktionen mit A_1 auf der linken Seite. Wir ersetzen, falls vorhanden, alle links-rekursiven A_1-Produktionen gemäß

dem Verfahren nach dem Beweis von Satz 5.5, wobei ein neues Nichtterminalzeichen Z_1 eingeführt wird. Wegen unserer Annahme besitzen die Z_1-Produktionen keine Terminalzeichen. Werden im Laufe des Verfahrens neue Nichtterminalzeichen $Z_1, Z_2, \ldots, Z_l$ für ein $l \in \mathbb{N}$ in dieser Reihenfolge eingeführt (wir werden sehen, daß $0 \leq l \leq n$ gilt), dann nehmen wir an, daß alle Nichtterminalzeichen in der Reihenfolge

$$Z_l, \ldots, Z_1, A_1, \ldots, A_n$$

geordnet sind. Der Begriff „bergauf" wird auf alle vorkommenden Produktionen mit den so geordneten Nichtterminalzeichen erweitert. Wir erhalten durch die oben angegebenen Ersetzungen eine ε-freie kontextfreie Grammatik G_1 mit folgenden Eigenschaften:

(a) $L(G_1) = L(G)$,
(b) die A_1-Produktionen und, falls vorhanden, Z_1-Produktionen in G_1 gehen bergauf,
(c) G_1 erfüllt die folgende Bedingung:
  (∗) Es befindet sich jeweils höchstens ein Terminalzeichen auf den rechten Seiten der Produktionen (und dann an der ersten Stelle).

Aufgrund der Konstruktion der neuen A_1-Produktionen nach dem Beweis von Satz 5.5 und der jenem Beweis folgenden Bemerkung ist die Bedingung (b) gültig. Wir stellen weiter fest, daß nur Produktionen $A_1 \to b$ mit $b \in V_T$ zu neuen Produktionen mit Terminalzeichen führen. Das sind dann Produktionen $A_1 \to bZ_1$. Die Bedingung (∗) ist also erfüllt.

Als nächstes betrachten wir A_2-Produktionen. Wir ersetzen, falls vorhanden, in jeder Produktion $A_2 \to A_1 w$ das Zeichen A_1 durch die rechten Seiten aller A_1-Produktionen von G_1. Für die so geänderte Grammatik bleibt die Bedingung (∗) erfüllt. Dabei können zusätzliche links-rekursive A_2-Produktionen entstehen. Dann ersetzen wir, falls vorhanden, unter Einführung eines neuen Nichtterminalzeichens Z_i alle links-rekursiven A_2-Produktionen gemäß dem Verfahren nach dem Beweis von Satz 5.5 ($i = 1$, falls im vorhergehenden Schritt keine links-rekursiven A_1-Produktionen ersetzt werden mußten, sonst $i = 2$). Wir erhalten eine ε-freie kontextfreie Grammatik G_2 mit den folgenden Eigenschaften:

(a) $L(G_2) = L(G)$,
(b) die A_1-, A_2- und, falls vorhanden, alle Z-Produktionen in G_2 gehen bergauf,
(c) G_2 erfüllt die Bedingung (∗).

Wir stellen fest, daß nur Produktionen $A_2 \to b$ oder $A_2 \to bw$ mit $b \in V_T$ und $w \in V_{N,1}^*$ zu neuen Produktionen mit Terminalzeichen führen, d.h. zu Produktionen $A_2 \to bZ_i$ oder $A_2 \to bwZ_i$. Somit ist die Bedingung (∗) erfüllt. Man beachte, daß für $i = 2$ auch Z_2-Produktionen $Z_2 \to Z_1 v'$ und $Z_2 \to Z_1 v' Z_2$ möglich sind. Aufgrund der festgelegten Reihenfolge aller Nichtterminalzeichen gehen diese jedoch, ebenso wie die A_1- und A_2-Produktionen, bergauf. Die Eigenschaft (b) ist also gültig.

Das Verfahren wird für die A_3-Produktionen usw. bis zu den A_n-Produktionen fortgesetzt, wobei die eben durchgeführten Überlegungen entsprechend gelten. Wir erhalten damit eine ε-freie kontextfreie Grammatik G_n mit den folgenden Eigenschaften:

(a) $L(G_n) = L(G)$,

(b) alle Produktionen in G_n gehen bergauf bezüglich der Reihenfolge $Z_l, \ldots, Z_1, A_1, \ldots, A_n$,
(c) G_n erfüllt die Bedingung $(*)$.

Wir kommen nun zur rücklaufenden Konstruktion. Die A_n-Produktionen sind schon in Greibachscher Normalform. Die A_{n-1}-Produktionen sind von der Form $A_{n-1} \to A_n w$ oder $A_{n-1} \to bw$ mit $w \in V_N^*$, $b \in V_T$. Wir ersetzen in den Produktionen der ersten Form A_n durch die rechte Seite aller A_n-Produktionen. Wegen der Bedingung $(*)$ sind dann auch die neuen A_{n-1}-Produktionen in Greibachscher Normalform. Im nächsten Schritt müssen bei den A_{n-2}-Produktionen entsprechende Ersetzungen für Produktionen der Form $A_{n-2} \to A_n w$ und $A_{n-2} \to A_{n-1} w$ durchgeführt werden.

Da alle Produktionen bergauf gehen, können wir das Verfahren fortsetzen bis zu den A_1-Produktionen und dann weiter von den Z_1-Produktionen bis zu den Z_l-Produktionen. Wir erhalten schließlich eine ε-freie kontextfreie Grammatik $\overline{G}$ mit $L(\overline{G}) = L(G)$ in Greibachscher Normalform. □

Für diesen Satz gibt es eine Vielzahl recht unterschiedlicher Beweise. Ein auf ganz anderen Methoden beruhender Beweis kann in [24], Satz VI.8.4 nachgelesen werden.

5.4 Kontextfreie Sprachen als deterministische Typ-1-Sprachen

Dieser Abschnitt würde seinen Sinn verlieren, wenn gezeigt werden könnte, daß deterministische und nichtdeterministische linear beschränkte Automaten äquivalent sind. Dies ist bekanntlich das offene lba-Problem. Mit $\mathfrak{L}_{1,d}$ bezeichnen wir die Familie der Sprachen, die durch deterministische linear beschränkte Automaten akzeptiert werden.

Satz 5.7 Es sei L eine kontextfreie Sprache. Dann existiert ein deterministischer linear beschränkter Automat B mit $L(B) = L$ (d.h.: $\mathfrak{L}_2 \subset \mathfrak{L}_{1,d}$).

Beweis: Ohne Beschränkung der Allgemeinheit können wir annehmen, daß L ε-frei ist. Anderenfalls würde man die Konstruktion für $L - \{\varepsilon\}$ durchführen und anschließend den Automaten B zu einen Automaten B' erweitern, dessen Anfangszustand gleichzeitig ein Endzustand ist. Es sei L eine kontextfreie Sprache mit $\varepsilon \notin L$. Nach Satz 5.4 existiert eine Grammatik $G = (V_N, V_T, X_0, F)$ in Chomskyscher Normalform mit $L(G) = L$. Es sei $w \in L$ mit $|w| = n$. Dann gibt es eine Linksableitung

$$X_0 = P_0 \Longrightarrow P_1 \Longrightarrow \ldots \Longrightarrow P_{2n-1} = w$$

der Länge $2n-1$. Dies ergibt sich daraus, daß jedes Terminalzeichen $a \in V_T$ in w durch eine Produktion $Y_a \to a$ entsteht. Für w sind n derartige Ableitungsschritte erforderlich. Da mit Hilfe von Produktionen $X \to YZ$ in jedem Ableitungsschritt die Anzahl der Nichtterminalzeichen um 1 erhöht wird, müssen wir $n-1$ solcher Ableitungsschritte durchführen, um von X_0 ausgehend die benötigten n Nichtterminalsymbole Y_a zu erzeugen. Zu jeder solchen Ableitung können wir einen Ableitungsbaum angeben, aus dem wir sofort eine Linksableitung für w konstruieren können.

Es sei $\text{card}(F) = k$. Die Produktionen von F bezeichnen wir durch

$$f_0, \ldots, f_{k-1}.$$

Jede Linksableitung läßt sich durch eine $(2n-1)$-stellige k-adische Zahl darstellen, bei der in jeder Stelle angegeben ist, welche Produktion bei dem entsprechenden Ableitungsschritt verwendet wird. Insgesamt gibt es k^{2n-1} verschiedene $(2n-1)$-stellige k-adische Zahlen.

Es wird ein deterministischer linear beschränkter Automat B konstruiert, der genau die Wörter $w \in L$ erkennt. Für $w = a_1 \ldots a_n$ können wir durch einen Vor- und Rücklauf des Kopfes eine Unterteilung des Eingabebandes in vier Spuren erreichen. Dabei hält die erste Spur das Eingabewort, während die zweite und dritte Spur eine $(2n-1)$-stellige k-adische Zahl aufnehmen können, die anfangs auf 0 gesetzt wird. Die Felder der Spur 4 sind jeweils mit einem „Blankzeichen“ $b \notin V_I$ beschrieben. Auf dieser Spur soll die Ableitung simuliert werden.

¢	a_1	a_2	a_3	$\cdots$		a_n	\$	Spur 1
	0	0	0	$\cdots$	0	0		Spur 2
	0	0	0	$\cdots$	0	b		Spur 3
	b	b	b	$\cdots$		b		Spur 4

B erzeugt nacheinander die Zahlen $m = 0, 1, \ldots, k^{2n-1} - 1$ als k-adische $(2n-1)$-stellige Zahlen

$$m = \alpha_{2n-1} + \alpha_{2n-2}k + \cdots + \alpha_1 k^{2n-2}.$$

Ist eine solche Zahl erzeugt, so beschreibt B das am weitesten links stehende Feld von Spur 4 mit X_0 und untersucht durch Simulation auf Spur 4, ob bei Verwendung der Produktionen $f_{\alpha_1}, \ldots, f_{\alpha_{2n-1}}$ in dieser Reihenfolge eine Linksableitung von w erhalten werden kann. Da es keine auslöschenden Produktionen gibt, ist der Platz auf Spur 4 ausreichend. Für die Simulation sieht der linear beschränkte Automat B auf den Spuren 2 und 3 nach, welche Produktionen er benutzen muß. Es wird das jeweils am weitesten links stehende Nichtterminalzeichen auf Spur 4, etwa X, durch die entsprechende rechte Seite des jeweiligen f_{α_i} ersetzt. Dies ist jedoch nur möglich, wenn die linke Seite von f_{α_i} gleich X ist und außerdem auf Spur 4 noch Platz vorhanden ist. Im anderen Fall wird die Simulation abgebrochen, es werden lauter Zeichen b auf Spur 4 eingetragen, die k-adische Zahl aus Spur 2 und 3 wird um 1 erhöht und die nächste Simulation gestartet. Konnte eine Simulation bis zum Ende durchgeführt werden, so wird das erzeugte Wort P mit w verglichen. Ist $P = w$, so geht B in einen Endzustand über. Anderenfalls wird, wie eben beschrieben, die Simulation der nächsten Ableitung vorbereitet und gestartet. Im Laufe der Arbeit akzeptiert B die Eingabe w, oder wir wissen, falls auch für $m = k^{2n-1} - 1$ keine Linksableitung für w gefunden wird, daß $w \notin L$ gilt. Dann hält B in einem Nichtendzustand.

Es ist anschaulich klar, daß die angegebenen Schritte durch einen deterministischen Automaten durchgeführt werden können. Einen solchen Automaten im Detail zu konstruieren, ist sehr aufwendig. Dies soll hier nicht geschehen. □

5.5 Das Iterationstheorem

Es sollen Sätze angegeben werden, die es unter Umständen ermöglichen, für eine vorgegebene Sprache festzustellen, daß sie nicht vom Typ 2 ist.

Definition 5.7 Es sei V_T ein Alphabet, $w \in V_T^*$, $\varphi = (v_1, \ldots, v_n) \in (V_T^*)^n$ mit $n \in \mathbb{N}$. Dann heißt φ eine *Faktorisierung* von w, wenn

$$w = v_1 \ldots v_n$$

gilt. Eine Zahl

$$i \in \mathbb{N},\ 1 \leq i \leq |w|,$$

heißt *Position* von w. Es sei K eine Menge von Positionen von w und φ eine Faktorisierung von w. Dann heißt

$$K_i = \{k \in K \mid |v_1 \ldots v_{i-1}| < k \leq |v_1 \ldots v_i|\}$$

die *Menge der markierten Positionen in* v_i.

$$K/\varphi = \{K_1, \ldots, K_n\}$$

wird als *Partition von* K bezeichnet. □

Es ist auch möglich, daß einige der K_i leer sind. K/φ ist also keine echte Partition in dem Sinn, wie dieser Begriff z.B. im Zusammenhang mit Äquivalenzrelationen benutzt wird.

Beispiel 5.6 (a) Es sei $w = a_1 \ldots a_n$, $a_i \in V_T$, $n \in \mathbb{N}$ und $K = \{1, \ldots, n\}$.
(α) Gilt $\varphi = (a_1, \ldots, a_n)$, so folgt $K/\varphi = \{K_i \mid i = 1, \ldots, n\}$ mit $K_i = \{i\}$.
(β) Gilt $\varphi = (a_1 \ldots a_n)$, so folgt $K/\varphi = \{K_1\} = \{K\}$.
(b) Es sei $p \in \mathbb{N}$. Wir setzen $w = a^p b^p c^p$, $\varphi = (a^p, b^p, c^p)$ und $K = \{p+1, \ldots, 2p\}$. Dann folgt $K_1 = K_3 = \emptyset$ und $K_2 = K$. □

Eine Menge von markierten Positionen kann in dem entsprechenden Wort durch eine geschweifte Klammer kenntlich gemacht werden, z.B. $a^p \underbrace{b^p}\ c^p$.

Der folgende Satz wird auch *Iterationstheorem* oder *Lemma von Ogden* genannt.

Satz 5.8 Es sei $G = (V_N, V_T, X_0, F)$ eine kontextfreie Grammatik mit $L = L(G)$. Dann existiert eine Zahl $p(G) \in \mathbb{N}$, so daß für alle $w \in L$ und alle Mengen K von markierten Positionen in w das folgende gilt: Ist $\mathrm{card}(K) \geq p(G)$, so gibt es eine Faktorisierung $\varphi = (v_1, v_2, v_3, v_4, v_5)$ von w mit folgenden Eigenschaften:

(1) Es existiert $A \in V_N$ mit
$X_0 \Longrightarrow^* v_1 A v_5$,
$A \Longrightarrow^* v_2 A v_4$,
$A \Longrightarrow^* v_3$.

(2) Für alle $q \in \mathbb{N}_0$ gilt $v_1 v_2^q v_3 v_4^q v_5 \in L$.
(3) Ist $K/\varphi = \{K_1, K_2, K_3, K_4, K_5\}$ so folgt
  (a) $K_1, K_2, K_3 \neq \emptyset$ oder $K_3, K_4, K_5 \neq \emptyset$ und
  (b) $\text{card}(K_2 \cup K_3 \cup K_4) \leq p(G)$.

Beweis: Wir bemerken zunächst, daß die Bedingung (3)(a) bedeutet, daß $v_3 \neq \varepsilon$ und $v_2 \neq \varepsilon$ oder $v_3 \neq \varepsilon$ und $v_4 \neq \varepsilon$ gilt und dort markierte Positionen sind.

Wir setzen $r = \max\{2, |\alpha| \mid (A, \alpha) \in F\}$, $m = \text{card}(V_N)$ und $p(G) = r^{2m+3}$. Die Konstante $p(G)$ hängt also nur von G ab. Die 2 in der Definition von r ist erforderlich, damit der Satz auch richtig bleibt, falls für alle Produktionen $(A, \alpha) \in F$ die Beziehung $|\alpha| \leq 1$ gilt. Die Wörter aus $L(G)$ hätten dann die Länge höchstens 1, es würde $p(G) = 0$ oder 1 gelten, aber eine Faktorisierung wie in der Satzformulierung angegeben wäre z.B. im Falle $p(G) = 1$ trotz $\text{card}(K) = 1 = p(G)$ für Wörter der Länge 1 nicht möglich.

Es sei $w = a_1 \ldots a_{|w|} \in L$, $a_i \in V_T$, $i = 1, \ldots, |w|$, und K eine Menge markierter Positionen in w mit $\text{card}(K) \geq p(G)$. Weiter sei T ein Ableitungsbaum für die Ableitung $X_0 \Longrightarrow_G^* w$. Dabei seien $y_1, \ldots, y_{|w|}$ die Blätter von T, die mit $a_1, \ldots, a_{|w|}$ bezeichnet sind. Wir definieren zwei Mengen von Knoten von T:

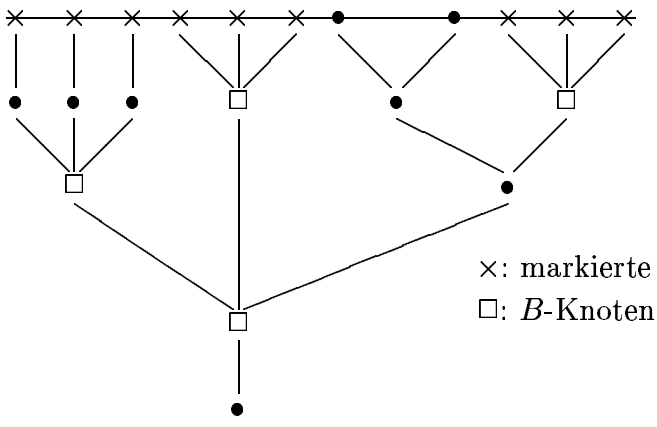
$$D = \{x \text{ Knoten in } T \mid \text{es existiert ein Pfad von } x \text{ nach } y_i,\ i \in K\},$$
$$B = \{x \text{ Knoten in } T \mid x \text{ besitzt Söhne } x_1, x_2 \in D,\ x_1 \neq x_2\}.$$

Die Knoten in D werden D-Knoten genannt, die Knoten in B entsprechend B-Knoten. Bei einem D-Knoten existiert ein Pfad zu einer markierten Position. Ein B-Knoten x besitzt zwei Söhne, die jeder einen Pfad zu einer markierten Position besitzen. Daher ist ein B-Knoten auch ein D-Knoten.

Behauptung 1: Falls jeder Pfad in T höchstens i B-Knoten besitzt, dann hat w höchstens r^i markierte Positionen in K.

Beweis von Behauptung 1: Eine maximale Anzahl von markierten Positionen erhält man, wenn jeder Sohn eines B-Knotens ein D-Knoten ist und so zu einer markierten Position führt. Da jeder Knoten, also auch jeder B-Knoten, höchstens r Söhne hat, folgt die Behauptung. Zum besseren Verständnis dieses Sachverhalts ist das folgende Beispiel angegeben.

$i = 2,\ r = 3$



□ (Behauptung 1)

Wir wählen einen Pfad $s = (s_0, \ldots, s_t)$ in T mit folgenden Eigenschaften:

(a) s_0 ist die Wurzel von T.
(b) s_t ist ein Blatt.
(c) s enthält eine maximale Anzahl von B-Knoten in bezug auf alle Pfade, die (a) und (b) erfüllen.

Behauptung 2: s enthält mindestens $2m + 3$ B-Knoten.

Beweis von Behauptung 2: Wir nehmen an, daß s höchstens $2m+2$ B-Knoten besitzt. Nach Wahl von s hat s eine maximale Anzahl von B-Knoten, so daß jeder Pfad in T von s_0 zu irgendeinem Blatt höchstens $2m + 2$ B-Knoten besitzt. Nach Behauptung 1 hat dann w höchstens r^{2m+2} markierte Positionen in K, was im Widerspruch zu $\text{card}(K) \geq p(G) = r^{2m+3}$ steht. □ (Behauptung 2)

Wir definieren eine Menge C_s von Knoten auf s durch die folgenden Bedingungen:

(1) $C_s \subset B$.
(2) Für alle $x \in C_s$ gilt $y \in C_s$ für jeden weiteren Nachkömmling $y \in B$ von x auf dem Pfad s.
(3) $\text{card}(C_s) = 2m + 3$.

Wir sehen, daß C_s die $2m + 3$ B-Knoten von s enthält, die dem Blatt s_t am nächsten sind.

Behauptung 3: Es sei $C_s = C_L \cup C_R$ für Mengen C_L und C_R. Dann folgt $\text{card}(C_L) \geq m + 2$ oder $\text{card}(C_R) \geq m + 2$.

Beweis von Behauptung 3: Würde die Folgerung nicht erfüllt sein, so ergäbe sich

$$\text{card}(C_s) \leq \text{card}(C_L) + \text{card}(C_R) \leq 2(m + 1) < 2m + 3,$$

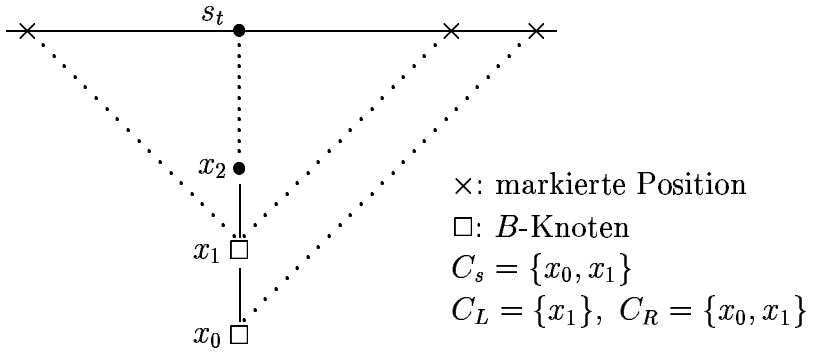
ein Widerspruch. □ (Behauptung 3)

Wir definieren

$C_L = \{\ x \in C_s \mid x$ besitzt einen Sohn y, der nicht auf s liegt, und es existiert ein Pfad von y zu einem Blatt, das einer markierten Position echt links von s_t entspricht$\}$,

$C_R = \{\ x \in C_s \mid x$ besitzt einen Sohn y, der nicht auf s liegt, und es existiert ein Pfad von y zu einem Blatt, das einer markierten Position echt rechts von s_t entspricht$\}$.

Zur Veranschaulichung betrachten wir das folgende Beispiel.



Behauptung 4: Es gilt $C_s = C_L \cup C_R$.

Beweis von Behauptung 4: Offensichtlich ist $C_L \cup C_R \subset C_s$. Umgekehrt sei $x \in C_s$. Dann ist x ein B-Knoten. Somit existieren wenigstens zwei Nachkömmlinge, die markierte Positionen in K sind. Einer von diesen kann eventuell s_t sein, der andere jedoch muß dann rechts oder links von s_t liegen. Es folgt $x \in C_R$ oder $x \in C_L$ und damit $x \in C_R \cup C_L$. □ (Behauptung 4)

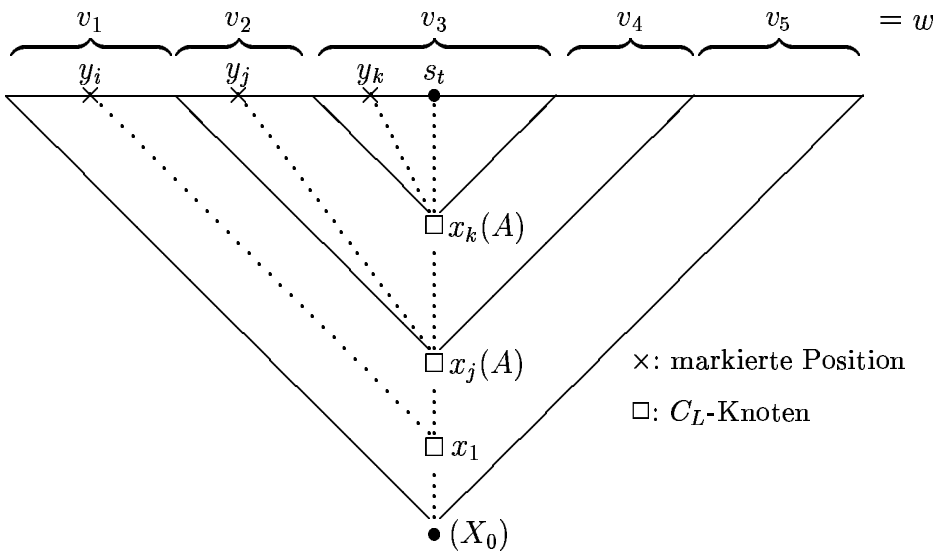
Nach Behauptung 3 sind zwei duale Fälle zu betrachten, und zwar

$$\text{card}(C_L) \geq m+2 \text{ oder } \text{card}(C_R) \geq m+2.$$

Ohne Beschränkung der Allgemeinheit nehmen wir an, daß $\text{card}(C_L) \geq m+2$ gilt. Wir schreiben

$$C_L = \{x_1, \ldots, x_{m'}\},$$

wobei x_{i+1} ein Nachkömmling von x_i ist für $i = 1, \ldots, m'-1$. Wegen $m' \geq m+2$ existieren $j, k \in \mathbb{N}$ mit $1 < j < k \leq m'$ sowie ein Zeichen $A \in V_N$, so daß x_j und x_k mit A markiert sind (siehe Abbildung).



Dann folgt

$$\begin{aligned} X_0 &\Longrightarrow^* v_1 A v_5 \text{ für geeignete } v_1, v_5 \in V_T^*, \\ A &\Longrightarrow^* v_2 A v_4 \text{ für geeignete } v_2, v_4 \in V_T^* \text{ und} \\ A &\Longrightarrow^* v_3 \text{ für ein geeignetes } v_3 \in V_T^*, \end{aligned}$$

wobei $w = v_1 v_2 v_3 v_4 v_5$ gilt. Die Aussage (1) des Satzes ist also erfüllt. Unter q-maliger ($q \in \mathbb{N}_0$) Iteration der zweiten Ableitung erhalten wir

$$X_0 \Longrightarrow^* v_1 A v_5 \Longrightarrow^* v_1 v_2^q A v_4^q v_5 \Longrightarrow^* v_1 v_2^q v_3 v_4^q v_5 \in L.$$

Damit ergibt sich die Aussage (2).

Wir betrachten jetzt die Knoten $x_1, x_j, x_k \in C_L$. Nach Definition von C_L existieren Nachkömmlinge y_i von x_1, y_j von x_j und y_k von x_k mit $i, j, k \in K$, so daß y_i echt links von y_j und y_j echt links von y_k liegt. Wir betrachten die Faktorisierung $\varphi = (v_1, v_2, v_3, v_4, v_5)$ und damit $K/\varphi = \{K_1, K_2, K_3, K_4, K_5)$. Wir erhalten

$$i \in K_1,\ j \in K_2,\ k \in K_3$$

und folglich auch $K_1 \neq \emptyset$, $K_2 \neq \emptyset$ und $K_3 \neq \emptyset$, so daß die Aussage (3)(a) des Satzes erfüllt ist.

Der Pfad s ist so gewählt, daß er eine maximale Anzahl von B-Knoten enthält. C_s besitzt $2m+3$ B-Knoten, x_1 ist ein Knoten von C_s. Folglich kann kein Pfad, der in x_1 startet, mehr als $2m+3$ B-Knoten haben. Anderenfalls ergäbe sich ein Widerspruch zur Maximalität von s und zur Definition von C_s. Jeder Pfad, der in x_j beginnt, kann zu einem Pfad erweitert werden, der in x_1 startet. Darum hat jeder in x_j beginnende Pfad höchstens $2m+3$ B-Knoten. Aus der Behauptung 1 folgt, daß der Teilbaum mit der Wurzel in x_j höchstens $r^{2m+3} = p(G)$ markierte Positionen in K besitzt. Dieser Teilbaum leitet $v_2v_3v_4$ ab. Wir erhalten $\text{card}(K_2 \cup K_3 \cup K_4) \leq p(G)$, also die Gültigkeit der Aussage (3)(b). □

Als Folgerung erhalten wir das sogenannte *Lemma von Bar-Hillel* oder *Pumping-Lemma*. Es ist leichter anzuwenden als das allgemeine Iterationstheorem. Es gibt allerdings Sprachen, für die man mit Hilfe des Iterationstheorem zeigen kann, daß sie nicht kontextfrei sind, bei denen jedoch das Lemma von Bar-Hillel versagt.

Satz 5.9 Es sei $G = (V_N, V_T, X_0, F)$ eine kontextfreie Grammatik mit $L(G) = L$. Dann existieren Zahlen $p, q \in \mathbb{N}$, so daß für alle $w \in L$ das folgende gilt: Ist $|w| \geq p$, so gibt es eine Faktorisierung $\varphi = (u_1, v_1, v, v_2, u_2)$ von w mit folgenden Eigenschaften:

(1) $|v_1vv_2| \leq q$,
(2) $v_1v_2 \neq \varepsilon$,
(3) für alle $i \in \mathbb{N}_0$ gilt $u_1v_1^ivv_2^iu_2 \in L$.

Beweis: Wir wenden Satz 5.8 an. Jede Position von w wird als markiert gewählt. Dann setzen wir $p = q = p(G)$. Aus Satz 5.8(3)(b) folgt (1), aus Satz 5.8(3)(a) ergibt sich (2), und aus Satz 5.8(2) erhalten wir (3). □

Ein direkter Beweis von Satz 5.9 (siehe z.B. [24], Satz II.6.4) ist viel kürzer. Man erhält dabei im allgemeinen verschiedene Werte p und q.

Beispiel 5.7 Die Sprache $L = \{a^nb^nc^n \mid n \in \mathbb{N}\}$ ist nicht kontextfrei. Wir nehmen das Gegenteil an. Es seien $p, q \in \mathbb{N}$ Zahlen mit den Eigenschaften aus Satz 5.9. Es wird ein $k \in \mathbb{N}$ mit $k > \frac{p}{3}$ gewählt und das Wort $w_1 = a^kb^kc^k$ betrachtet. Nach Satz 5.9 existiert eine Faktorisierung mit

$$w_1 = u_1v_1vv_2u_2,\ v_1v_2 \neq \varepsilon,\ |v_1vv_2| \leq q,\ w_i = u_1v_1^ivv_2^iu_2 \in L \text{ für alle } i \in \mathbb{N}_0.$$

v_1 und v_2 enthalten jeweils höchstens eine Sorte der Zeichen a, b, c, da sich anderenfalls schon bei w_2 eine falsche Reihenfolge der Symbole a, b, c ergäbe. Das bedeutet, daß

mindestens ein Zeichen $x \in \{a, b, c\}$ weder in v_1 noch in v_2 vorkommt. Dieses Zeichen x ist in jedem w_i gleich häufig. Dies steht jedoch aufgrund der Gestalt von L im Widerspruch dazu, daß wegen $v_1 v_2 \neq \varepsilon$ die Wörter w_i mit wachsendem i immer länger werden. □

Beispiel 5.8 Es sei

$$L = \{a^n bc \mid n \in \mathbb{N}_0\} \cup \{a^p b a^n c a^n \mid p \text{ prim}, n \in \mathbb{N}_0\}.$$

Wir nehmen an, daß L kontextfrei ist. Es sei $p(G)$ gemäß Satz 5.8 bestimmt. Es werde eine Primzahl $p > p(G)$ gewählt und das Wort

$$w = \underbrace{a^p} ba^p ca^p \in L$$

betrachtet. Nach Satz 5.8 existiert eine Faktorisierung $\varphi = (v_1, v_2, v_3, v_4, v_5)$ von w, wobei sich

$$\text{(a) in jeweils } v_1, v_2 \text{ und } v_3 \text{ oder (b) in jeweils } v_3, v_4 \text{ und } v_5$$

markierte Positionen befinden.

Im Fall (b) sind v_2 und v_4 vollständig im markierten Teilwort a^p enthalten. Es gelte etwa $v_2 v_4 = a^k$ für ein geeignetes k, $1 \leq k \leq p(G)$. Nach Satz 5.8(2) folgt

$$w_i = v_1 v_2^i v_3 v_4^i v_5 = a^{p-k+ik} b a^p c a^p \in L \text{ für alle } i \in \mathbb{N}_0.$$

Speziell für $i = p + 1$ gilt

$$p - k + ik = p + pk = p(k + 1),$$

d.h., $p(k + 1)$ ist keine Primzahl, also gilt $w_{p+1} \notin L$, ein Widerspruch.

Da b in keiner markierten Position liegt und außerdem nicht iteriert werden darf, ist im Fall (a) b ein Teilwort von v_3 oder v_5. Folglich ist v_2 ein Teilwort des markierten a^p. Wegen $v_2 \neq \varepsilon$ ergibt sich ein Widerspruch wie im Fall (b). Insgesamt schließen wir, daß L nicht kontextfrei ist.

Im folgenden zeigen wir, daß zum Nachweis, daß L nicht kontextfrei ist, Satz 5.9, also das Lemma von Bar-Hillel, nicht anwendbar ist. Wir wählen eine beliebige Zahl $p_0 = q \in \mathbb{N}$ mit $p_0 = q \geq 3$. Wir zeigen, daß damit die Aussagen von Satz 5.9 erfüllt sind (mit p_0 anstelle von p). Es gibt in L zwei Typen von Wörtern. Wir betrachten zunächst ein Wort der Gestalt $w = a^m bc \in L$ mit $|w| \geq q$. Dann existiert die Faktorisierung

$$u_1 = a^{m-1},\ v_1 = a,\ v = bc,\ v_2 = u_2 = \varepsilon.$$

Es gilt $|v_1 v v_2| = |abc| \leq q$, $v_1 v_2 = a \neq \varepsilon$ und $u_1 v_1^i v v_2^i u_2 = a^{m-1+i} bc \in L$ für alle $i \in \mathbb{N}_0$. Die Bedingungen von Satz 5.9 sind erfüllt.

Weiter sei w ein Wort der Gestalt $w = a^p b a^n c a^n$ mit $|w| \geq q$, einer Primzahl p und $n \geq 1$. Der Fall $n = 0$ ist schon zuvor behandelt worden. Wir erhalten eine Faktorisierung

$$u_1 = a^p b a^{n-1},\ v_1 = a,\ v = c,\ v_2 = a,\ u_2 = a^{n-1}$$

von w mit $|v_1vv_2| = |aca| = 3 \leq q$, $v_1v_2 \neq \varepsilon$ und $w_i = u_1v_1^ivv_2^iu_2 = a^pba^{n-1}a^ica^ia^{n-1} \in L$ für alle $i \in \mathbb{N}_0$. Man beachte, daß sich speziell für $n = 1$ und $i = 0$ die Beziehung $a^pbc \in \{a^nbc \mid n \in \mathbb{N}_0\}$ ergibt, die erste Menge der Vereinigung also wichtig ist, um zu zeigen, daß die Bedingungen von Satz 5.9 erfüllt sind. □

5.6 Deterministische Typ-2-Sprachen

Wir sind jetzt in der Lage, den Beweis von Satz 3.5 nachzuholen. Wir erinnern daran, daß wir nach Definition 3.9

$$P_i(L) = \{w \mid w \in L, \text{ genau } i-1 \text{ Präfixe von } w \text{ gehören zu } L\}$$

gesetzt haben.

Satz 5.10 (Satz 3.5) Es existieren Typ-2-Sprachen, die keine deterministischen Typ-2-Sprachen sind.

Beweis: Es ist $L = \{wsp(w) \mid w \in \{a,b\}^*\}$ eine Typ-2-Sprache, da sie von der kontextfreien Grammatik $G = (\{X\}, \{a,b\}, X, F)$ mit $F = \{X \to aXa, X \to bXb, X \to \varepsilon\}$ erzeugt wird. Es werde angenommen, daß L eine deterministische Typ-2-Sprache ist. Nach Satz 3.7 folgt, daß $P_i(L)$ für alle $i \in \mathbb{N}$ eine deterministische Typ-2-Sprache ist. Es sei R eine beliebige reguläre Sprache. Nach Satz 3.6 ist $P_i(L) \cap R$ eine deterministische Typ-2-Sprache. Wir wählen speziell $i = 4$ und $R = \{ab\}^+\{ba\}^+\{ab\}^+\{ba\}^+$ und setzen $L' = P_4(L) \cap R$. L' besteht aus genau den Wörtern $wsp(w) \in \{a,b\}^+$, die in R liegen und genau ein echtes Präfix ($\neq \varepsilon$, $\neq wsp(w)$) der Form $usp(u)$ enthalten. In der Mitte der Wörter $wsp(w)$ und $usp(u)$ müssen jeweils 2 gleiche Symbole aufeinanderfolgen. In R gibt es nur 3 derartige Stellen. Die dritte Stelle kommt nicht in Frage, da sonst links davon bb und aa aufgetreten wären, die in der hinteren Hälfte des Wortes dann nicht mehr vorkommen. Die zweite Stelle liefert das Wort $wsp(w)$ selbst. Folglich gilt mit der ersten Stelle $usp(u) = (ab)^n(ba)^n$ für alle $n \in \mathbb{N}$. $n = 0$ wäre kein echtes Präfix. Wir erhalten also

$$L' = \{(ab)^n(ba)^{n+j}(ab)^{n+j}(ba)^n \mid n \in \mathbb{N}, j \in \mathbb{N}_0\}.$$

Unter der Annahme, daß L' kontextfrei ist, seien $p, q \in \mathbb{N}$ die nach Satz 5.9 bestimmten Zahlen. Wir setzen $n = \max\{p,q\}$ und betrachten das Wort

$$w' = (ab)^n(ba)^n(ab)^n(ba)^n = w_1w_2w_3w_4 \text{ mit } w_1 = w_3 = (ab)^n, w_2 = w_4 = (ba)^n.$$

Es sei $w' = u_1v_1vv_2u_2$ eine Faktorisierung. v_1 oder v_2 dürfen nicht aa oder bb enthalten, folglich sind v_1 und v_2 Teilwörter von w_1, w_2, w_3 oder w_4. Wird in w_1 oder w_4 iteriert, dann müssen wegen $w' = wsp(w)$ sowohl w_1 als auch w_4 iteriert werden, und es ergeben sich Wörter $(ab)^{n+k}(ba)^n(ab)^n(ba)^{n+k} \in L'$ für geeignete $k \in \mathbb{N}$, ein Widerspruch. Wird in w_2 oder w_3 iteriert, dann müssen w_2 und w_3 iteriert werden. Bei 0-maliger Iteration erhalten wir ein Wort $(ab)^n(ba)^k(ab)^k(ba)^n \in L'$ mit $k \leq n-1$, ebenfalls ein Widerspruch. Wir schließen, daß L' und damit auch L keine deterministische Typ-2-Sprache sein kann. □

5.7 Zwei Entscheidbarkeitsfragen

Aufgrund der Churchschen These geben wir in diesem Abschnitt als auch in späteren Kapiteln Algorithmen in intuitiver Form an.

Satz 5.11 Es sei G eine kontextfreie Grammatik. Dann existiert jeweils ein (effektiver) Algorithmus, der entscheidet, ob
(a) $L(G) = \emptyset$,
(b) $L(G)$ unendlich
ist.

Beweis: Wir beweisen zunächst (a). Es sei $G = (V_N, V_T, X_0, F)$ gegeben. Wir definieren

$$U_1 = \{X \in V_N \mid \bigvee_{w \in V_T^*} (X, w) \in F\},$$
$$U_{i+1} = \{X \in V_N \mid \bigvee_{P \in (U_i \cup V_T)^*} (X, P) \in F\} \text{ für } i \in \mathbb{N}.$$

Dann ist U_i eine wachsende Folge von Teilmengen von V_N. Somit existiert ein $k \in \mathbb{N}$ mit $U_k = U_{k+\nu}$ für alle $\nu = 1, 2, \ldots$. U_k kann effektiv konstruiert werden. Offenbar gilt

$$L(G) = \emptyset \iff X_0 \notin U_k,$$

was sofort überprüft werden kann.

Zum Beweis von (b) nehmen wir an, daß zur Grammatik G die Zahlen $p, q \in \mathbb{N}$ gemäß Satz 5.9 gewählt sind. Wir zeigen die folgende Behauptung:

$$\text{card}(L(G)) = \infty \iff \bigvee_{w' \in L(G)} p < |w'| \leq p + q.$$

Ist $|w'| > p$, so kann nach Satz 5.9(3) innerhalb des Wortes w' iteriert werden. Damit erhalten wir unendlich viele Wörter von $L(G)$. Ist umgekehrt $L(G)$ unendlich, so existiert $w \in L(G)$ mit

$$|w| > p + q,\ w = u_1 v_1 v v_2 u_2,\ v_1 v_2 \neq \varepsilon,\ |v_1 v v_2| \leq q,\ u_1 v u_2 \in L(G).$$

Die letzte Beziehung folgt wegen Satz 5.9(3) mit $i = 0$. Aufgrund von $|w| > p+q$ und $|v_1 v v_2| \leq q$ folgt $p < |u_1 v u_2| < |w|$. Falls $|u_1 v u_2| \leq p + q$ gilt, setzen wir $w' = u_1 v u_2$, und die Behauptung ist bewiesen. Anderenfalls wiederholen wir das Verfahren mit $u_1 v u_2$ anstelle von w. Nach endlich vielen Schritten erhalten wir ein Wort $w' \in L(G)$ mit $p < |w'| \leq p + q$.

Zur Grammatik G wird nach Satz 5.2(a) eine ε-freie kontextfreie Grammatik G' mit $L(G') = L(G) - \{\varepsilon\}$ konstruiert. Nach Satz 5.4 kann eine kontextfreie Grammatik G_1 in Chomskyscher Normalform mit $L(G_1) = L(G')$ angegeben werden. G und G_1 erzeugen dieselben Wörter w mit $p < |w| \leq p+q$. Da G_1 in Chomskyscher Normalform ist, existieren nur endlich viele Ableitungsbäume von G_1 mit $> p$ und $\leq p+q$ Blättern.

Diese entsprechen endlich vielen Wörtern $P \in V^*$ (Satzformen) mit $p < |P| \leq p+q$, die mit G_1 erzeugbar sind. Es ist nachprüfbar, ob darunter ein Wort aus V_T^*, also ein Wort aus $L(G)$, vorkommt. Nach der oben stehenden Behauptung kann so entschieden werden, ob $L(G)$ unendlich ist oder nicht. □

Im Beweis von Teil (a) des Satzes muß bei der Definition von U_{i+1} die Bedingung $P \in (U_i \cup V_T)^*$ gestellt werden, $P \in U_i^*$ würde zum Fehler führen. Dies erkennen wir zum Beispiel an einer Grammatik mit den Produktionen

$$X_0 \to aXb,\ X \to a,$$

mit der man richtig $U_1 = \{X\}$ und $U_2 = \{X, X_0\} = U_3 = \ldots$ erhält. Im anderen Fall ergäbe sich fälschlich $U_2 = \{X\} = U_3 = \ldots$.

5.8 Abschlußeigenschaften von kontextfreien Sprachen

Satz 5.12 Die Sprachfamilie $\mathfrak{L}_2$ ist nicht abgeschlossen unter
(a) Durchschnittsbildung,
(b) Komplementbildung.

Beweis: Zum Beweis von (a) betrachten wir die kontextfreien Grammatiken

$$G_1 = (\{X_0, X_1\}, \{a, b, c\}, X_0, \{X_0 \to X_0c, X_0 \to X_1c, X_1 \to ab, X_1 \to aX_1b\}),$$
$$G_2 = (\{X_0, X_1\}, \{a, b, c\}, X_0, \{X_0 \to aX_0, X_0 \to aX_1, X_1 \to bc, X_1 \to bX_1c\}).$$

Es folgt

$$L(G_1) = \{a^n b^n c^m \mid n, m \in \mathbb{N}\},$$
$$L(G_2) = \{a^m b^n c^n \mid n, m \in \mathbb{N}\}$$

und damit

$$L(G_1) \cap L(G_2) = \{a^n b^n c^n \mid n \in \mathbb{N}\}.$$

Nach Beispiel 5.7 ist dieser Durchschnitt nicht kontextfrei.

Wäre $\mathfrak{L}_2$ unter Komplementbildung abgeschlossen, so würde

$$L(G_1) \cap L(G_2) = \complement((\complement L(G_1)) \cup (\complement L(G_2))) \in \mathfrak{L}_2$$

gelten, ein Widerspruch. □

Satz 5.13 Es sei L eine kontextfreie und R eine reguläre Sprache. Dann gilt:
(a) Es ist $L \cap R$ kontextfrei.
(b) Falls L eindeutig ist, so ist auch $L \cap R$ eindeutig.

Beweis: Ohne Beschränkung der Allgemeinheit gelte $L, R \subset V_T^*$, wobei V_T ein Alphabet ist. Wir nehmen zunächst an, daß $\varepsilon \in L$ gilt. Dann existiert nach Satz 5.2 (a) eine ε-freie kontextfreie Grammatik G_1 mit $L(G_1) = L - \{\varepsilon\}$. Ist L außerdem eindeutig, so gibt es nach Definition 5.3(b) eine eindeutige Grammatik G mit $L(G) = L$. Bei der Konstruktion nach Satz 5.2(a) ist die sich aus G ergebende Grammatik G_1 ebenfalls eindeutig. Wir sehen dies wie folgt ein. Für alle $w \in L(G)$, $w \neq \varepsilon$, gibt es in G genau einen Ableitungsbaum. Die Konstruktion von G_1 ist so gegeben, daß wir aus einem Ableitungsbaum T von w gemäß G einen Ableitungsbaum T_1 von w gemäß G_1 dadurch erhalten, daß wir alle Teilbäume von T, die zum leeren Wort führen, entfernen. Diese Konstruktion liefert einen eindeutig bestimmten Ableitungsbaum T_1. Andere Ableitungsbäume von w kann es nach der Konstruktion von G_1 aus G nicht geben. Folglich ist $L - \{\varepsilon\}$ eindeutig.

Ist $\varepsilon \in R$, so können wir ebenfalls die Konstruktion nach Satz 5.2(a) verwenden, da bei diesem Vorgehen eine Typ-3-Grammatik eine solche bleibt. Es existiert also eine Typ-3-Grammatik G_2 mit $L(G_2) = R - \{\varepsilon\}$.

Wir werden eine Typ-2-Grammatik mit Anfangssymbol X_0 konstruieren, die die Sprache $(L-\{\varepsilon\}) \cap (R-\{\varepsilon\})$ erzeugt. Es wird gezeigt, daß der Durchschnitt eindeutig ist, falls $L-\{\varepsilon\}$ eindeutig ist. Falls $\varepsilon \in L \cap R$ gilt, werden ein neues Anfangssymbol X_0' sowie Produktionen $X_0' \to \varepsilon$ und $X_0' \to X_0$ zur Grammatik hinzugefügt. Diese neue Grammatik erzeugt $L \cap R$.

Im folgenden können wir also ohne Beschränkung der Allgemeinheit annehmen, daß L und R ε-freie Sprachen sind. R werde durch einen endlichen erkennenden Automaten $E = (S, V_T, \delta, s_0, S_1)$ akzeptiert. Es sei $S_1 = \{s^1, \ldots, s^h\}$. Dann definieren wir endliche erkennende Automaten $E_i = (S, V_T, \delta, s_0, \{s^i\})$ für alle i, $i = 1, \ldots, h$. Wir setzen $R_i = L(E_i)$. Dann folgt $R = \bigcup_{i=1}^{h} R_i$. Diese Vereinigung ist disjunkt, da die Automaten E_i deterministisch sind. Wir erhalten

$$(*) \quad L \cap R = (L \cap R_1) \cup \ldots \cup (L \cap R_h).$$

Auch dies ist eine disjunkte Vereinigung. Für ein festes i, $1 \leq i \leq h$, wird $L \cap R_i$ durch die Grammatik

$$G_i = ((V_N \cup V_T) \times S \times S, V_T, (X_0, s_0, s^i), F')$$

erzeugt, wobei F' definiert ist durch

(α) $(X, s, s') \to (y_1, s, \bar{s}_1)(y_2, \bar{s}_1, \bar{s}_2) \ldots (y_n, \bar{s}_{n-1}, s')$ für $s, \bar{s}_1, \bar{s}_2, \ldots, \bar{s}_{n-1}, s' \in S$,
$(X, y_1 \ldots y_n) \in F$, $y_j \in (V_N \cup V_T)$, $j = 1, \ldots, n$,
(β) $(a, s, s') \to a$ für $a \in V_T$, $\delta(s, a) = s'$.

In den Nichtterminalzeichen bleiben alle Ableitungen gemäß G durch die erste Komponente von (α) erhalten, während die anderen Komponenten bei einer Ableitung gemäß G_i eine Folge von Zuständen liefern mit der aus (α) ersichtlichen Nebenbedingung. Aus der Wahl des Anfangssymbols und (β) folgt, daß ein $w \in V_T^+$ genau dann von G_i erzeugt wird, wenn w einen Zustandsübergang von s_0 nach s^i in E_i bewirkt. Somit

gilt $L(G_i) = L \cap R_i$. Wegen $(*)$ und Satz 2.3 ist auch $L \cap R$ kontextfrei. Damit ist (a) bewiesen.

Weiter sei L eindeutig, L werde also durch eine eindeutige kontextfreie Grammatik G erzeugt. Wir zeigen zunächst die folgende Behauptung:

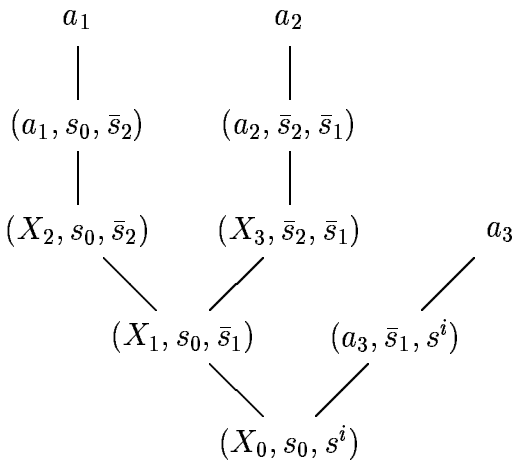
> Es seien $L_1, \ldots, L_n$ eindeutige Sprachen mit $L_i \cap L_j = \emptyset$ für $i \neq j$. Dann ist $\bigcup_{i=1}^{n} L_i$ eindeutig.

Zum Beweis betrachten wir eindeutige Sprachen L und L'. Ohne Beschränkung der Allgemeinheit seien sie Sprachen über demselben Alphabet V_T. Dann existieren eindeutige kontextfreie Grammatiken $G = (V_N, V_T, X_0, F)$ und $G' = (V'_N, V_T, X'_0, F')$ mit $L = L(G)$ und $L' = L(G')$. Ohne Beschränkung der Allgemeinheit gelte $V_N \cap V'_N = \emptyset$. Dann können wir die Grammatik

$$\bar{G} = (V_N \cup V'_N \cup \{\bar{X}_0\}, V_T, \bar{X}_0, F \cup F' \cup \{\bar{X}_0 \to X_0, \bar{X}_0 \to X'_0\})$$

mit $L(\bar{G}) = L \cup L'$ konstruieren, die offenbar eindeutig ist. Durch endliche Wiederholung dieser Konstruktion wird die Behauptung für eine beliebige Anzahl von eindeutigen Sprachen bewiesen.

Aufgrund dieser Behauptung und der Gleichung $(*)$ müssen wir nur noch beweisen, daß die Sprachen $L \cap R_i$, $i = 1, \ldots, h$ eindeutig sind. Dazu reicht es zu zeigen, daß die Grammatik G_i aus dem Beweis von (a) eindeutig ist. Es sei $w = a_1 \ldots a_m \in L \cap R_i$, $a_j \in V_T$, $j = 1, \ldots, m$. Wir betrachten einen Ableitungsbaum von w in G_i. Zur Illustration geben wir für $w = a_1 a_2 a_3$ ein Beispiel an:



In der ersten Komponente erfolgt die Ableitung wie in der eindeutigen Grammatik G. Der Ableitungsbaum hat dieselbe Gestalt wie der in G unter zusätzlicher Verwendung von Produktionen $(a, s, s') \to a$. Höchstens durch die Bezeichnung der zweiten und dritten Komponente der Knoten dieses Erzeugungsbaums könnte die Eindeutigkeit von G verloren gehen. Es gilt jedoch $w \in R_i$ genau dann, wenn eine Folge $s_0, \bar{s}_1, \ldots, \bar{s}_{m-1}$ von Zuständen existiert mit $\delta(s_0, a_1) = \bar{s}_1$, $\delta(\bar{s}_1, a_2) = \bar{s}_2 \ldots$, $\delta(\bar{s}_{m-1}, a_m) = s^i$. Da E_i

deterministisch ist, sind diese Zustände für die Ableitung von $w \in L \cap R$ eindeutig festgelegt. Das bedeutet, daß auch die Bezeichnungen der zweiten und dritten Komponente der Knoten im Ableitungsbaum eindeutig bestimmt sind. Es gibt also genau einen Ableitungsbaum für jedes $w \in L \cap R$. Somit ist $L \cap R$ eindeutig. □

Satz 5.14 Es sei L eine kontextfreie und R eine reguläre Sprache. Dann gilt:
(a) Es ist $L - R$ kontextfrei.
(b) Falls L eindeutig ist, so ist auch $L - R$ eindeutig.

Beweis: Nach Satz 1.2 und Satz 3.3 ist das Komplement einer regulären Sprache regulär. Es gilt

$$L - R = L \cap (\complement R).$$

Nach Satz 5.13 folgt daraus die Behauptung. □

Beispiel 5.9 Die Programmiersprache Algol 68 (und entsprechend andere Programmiersprachen) ist nicht kontextfrei. Es sei L die Menge der syntaktisch und semantisch richtigen Algol 68-Programme. Wir nehmen an, daß L kontextfrei ist und betrachten den Durchschnitt von L mit einer regulären Menge, nämlich

$$M = L \cap (\{\textbf{begin}\}\{\textbf{real}\}\{x\}^*\{;\}\{x\}^*\{:=\}\{x\}^*\{\textbf{end}\}).$$

Nach Satz 5.13 ist M kontextfrei. Wegen der semantischen Regel, daß jede Variable, die in einem gültigen Algol-Programm vorkommt, zuvor deklariert sein muß, gilt

$$M = \{\textbf{begin real } x^i;\ x^j := x^k\textbf{end} \mid i = j = k,\ i \in \mathbb{N}\}.$$

Abgekürzt können wir M als

$$M = \{brx^isx^iex^id \mid i \in \mathbb{N}\}$$

schreiben. Wir wählen $p, q \in \mathbb{N}$ gemäß Satz 5.9 und betrachten $w_1 = brx^psx^pex^pd$ mit $|w_1| > p$. Nach Satz 5.9 gilt

$$w_1 = u_1v_1vv_2u_2,\ v_1v_2 \neq \varepsilon, |v_1vv_2| \leq q,\ w_i = u_1v_1^ivv_2^iu_2 \in L \text{ für alle } i \in \mathbb{N}_0.$$

Die Symbole b, r, s, e, d dürfen nicht iteriert werden und liegen daher nicht in v_1 oder v_2. Folglich sind v_1 und v_2 Teilwörter von x^p. An höchstens zwei Stellen, mindestens jedoch an einer, wird iteriert. Folglich liegen in M Wörter mit unterschiedlich vielen x an den verschiedenen Stellen, ein Widerspruch. □

Kapitel 6

Weitere Charakterisierung von Typ-2- und Typ-0-Sprachen

In diesem Kapitel geben wir Charakterisierungen von Typ-2- und Typ-0-Sprachen als homomorphe Bilder geeigneter anderer Sprachen von eingeschränkter Form an.

6.1 Homomorphe Charakterisierungen von Typ-2-Sprachen

Definition 6.1 Für $n \in \mathbb{N}$ sei $V_n = \{a_1, a_1', \ldots, a_n, a_n'\}$ ein Alphabet. Damit wird eine kontextfreie Grammatik

$$G_n = (\{X\}, V_n, X, \{X \to \varepsilon, X \to XX, X \to a_1 X a_1', \ldots, X \to a_n X a_n'\})$$

definiert. Die davon erzeugte Sprache $D_n = L(G_n)$ heißt *Dyck-Sprache*. Speziell ist $D_0 = \{\varepsilon\}$. $\square$

Man erkennt vielleicht, daß D_n die Ausdrücke enthält, in denen n verschiedene Arten von Klammern „richtig“ zusammengestellt sind. Genauer gilt

Satz 6.1 Es sei D_n die Dyck-Sprache für $n \in \mathbb{N}_0$. Dann hat D_n die folgenden Eigenschaften:

(a) $(w_1 \in D_n \wedge w_2 \in D_n) \implies w_1 w_2 \in D_n$,

(b) $w \in D_n \implies \bigwedge_{i=1,\ldots,n} a_i w a_i' \in D_n$.

(c) $(w \in D_n \wedge w \neq \varepsilon) \implies \bigvee_{i,k \in \{1,\ldots,n\}} \bigvee_{w_1, w_2, \bar{w}_1, \bar{w}_2 \in D_n} w = a_i w_1 a_i' w_2 = \bar{w}_1 a_k \bar{w}_2 a_k'$,

(d) $a_i a_i' w \in D_n \implies w \in D_n$,

(e) $(w_1 w w_2 \in D_n \wedge w \in D_n) \iff (w_1 w_2 \in D_n \wedge w \in D_n)$.

Beweis: Die Eigenschaften (a) bis (d) folgen unmittelbar aus Definition 6.1. Für den Beweis von (e) betrachten wir $w_1 w w_2 \in D_n$ mit $w \in D_n$. Wegen $w \in D_n$ gibt es eine

Ableitung $X \Longrightarrow^* w_1 X w_2 \Longrightarrow^* w_1 w w_2$. Mit Hilfe der Produktion $X \to \varepsilon$ erhalten wir damit auch eine Ableitung $X \Longrightarrow^* w_1 X w_2 \Longrightarrow^* w_1 w_2$, also $w_1 w_2 \in D_n$.

Für die umgekehrte Richtung gelte $w_1 w_2, w \in D_n$. Der Fall $w_2 = \varepsilon$ ist durch (a) bewiesen. Ist $w_2 = a_i' v_2$, so gilt $w_1 = u_1 a_i u_2$, wobei a_i die korrespondierende „öffnende Klammer" von a_i' ist. Das Klammerpaar (a_i, a_i') wird mit Hilfe der Produktion $X \to a_i X a_i'$ eingeführt. Es existiert somit eine Ableitung

$$X \Longrightarrow^* u_1 X v_2 \Longrightarrow u_1 a_i X a_i' v_2 \Longrightarrow^* u_1 a_i u_2 a_i' v_2 = w_1 w_2,$$

wobei aufgrund von $X \Longrightarrow^* u_2$ die Beziehung $u_2 \in D_n$ gelten muß. Wegen $X \to XX$ und $w \in D_n$ können wir daraus eine Ableitung

$$\begin{aligned} X &\Longrightarrow^* u_1 X v_2 \Longrightarrow u_1 a_i X a_i' v_2 \Longrightarrow u_1 a_i X X a_i' v_2 \\ &\Longrightarrow^* u_1 a_i u_2 X a_i' v_2 \Longrightarrow^* u_1 a_i u_2 w a_i' v_2 = w_1 w w_2 \end{aligned}$$

konstruieren. Folglich gilt $w_1 w w_2 \in D_n$.

Gilt $w_2 = a_i v_2$, so erhalten wir $w_2 = a_i u_1 a_i' u_2$ mit korrespondierendem Klammerpaar (a_i, a_i'). Es existiert also eine Ableitung

$$X \Longrightarrow^* w_1 X u_2 \Longrightarrow w_1 a_i X a_i' u_2 \Longrightarrow^* w_1 a_i u_1 a_i' u_2 = w_1 w_2$$

mit $u_1 \in D_n$. Daraus ergibt sich eine Ableitung

$$X \Longrightarrow^* w_1 X u_2 \Longrightarrow w_1 X X u_2 \Longrightarrow w_1 X a_i X a_i' u_2 \Longrightarrow^* w_1 w a_i u_1 a_i' u_2 = w_1 w w_2.$$

Es folgt $w_1 w w_2 \in D_n$. $\square$

Der folgende Satz dient als Grundlage für die beiden Hauptergebnisse dieses Abschnitts, nämlich für Satz 6.4 und Satz 6.5. Seine sehr algebraische Aussage wird im anschließenden Beispiel 6.1 veranschaulicht.

Satz 6.2 Es sei V_T ein Alphabet und $L \subset V_T^+$ eine kontextfreie Sprache. Dann existiert ein Alphabet V_n wie in Definition 6.1 mit den disjunkten Teilmengen $V = \{a_1, \ldots, a_n\}$ und $V' = \{a_1', \ldots, a_n'\}$ und weiter ein Symbol $S \in V$ sowie ein Monoidhomomorphismus $\varphi : V_T^* \to \mathfrak{P}'(V_n^*)$ (wobei $\mathfrak{P}'(V_n^*)$ die Menge der endlichen Teilmengen von V_n^* ist), so daß die Äquivalenz

$$w \in L \iff \varphi(w) \cap \{S'\} D_n \neq \emptyset$$

erfüllt ist. Außerdem gilt $\varphi(x) \subset V' V^*$ für alle $x \in V_T$.

Beweis: Zunächst muß gezeigt werden, daß für ein Alphabet V_n die Menge $\mathfrak{P}'(V_n^*)$ ein Monoid ist. Dazu definieren wir für endliche Mengen $U_1, U_2 \subset V_n^*$ die Multiplikation durch

$$U_1 \cdot U_2 = \{u_1 u_2 \mid u_1 \in U_1, u_2 \in U_2\}$$

und wählen $\{\varepsilon\} \subset V_n^*$ als Einselement. Offenbar ist dann $\mathfrak{P}'(V_n^*)$ ein Monoid.

Nach Satz 5.6 existiert eine Grammatik $G = (V, V_T, X_0, F)$ in Greibachscher Normalform, die L erzeugt. Durch das Nichtterminalalphabet V der Grammatik ist das Alphabet $V_n = V \cup V'$ definiert. Ohne Beschränkung der Allgemeinheit können wir annehmen, daß es keine überflüssigen Symbole in V_T gibt, d.h., für jedes $x \in V_T$ existieren $A \in V$, $u \in V^*$ mit $(A, xu) \in F$. Wir geben den Homomorphismus φ durch

$$\varphi(x) = \{A'sp(u) \mid (A, xu) \in F\} \text{ für alle } x \in V_T$$

an. Es ist $\varphi(x) \subset V'V^*$ für alle $x \in V_T$ erfüllt. Durch Induktion über $|w|$ zeigen wir:

$$\bigwedge_{A \in V, w \in V_T^*, u \in V^*} (A \Longrightarrow^*_{\text{links}} wu \iff \bigvee_{m \in \varphi(w)} msp(u') \in \{A'\}D_n).$$

Dabei wird $\varepsilon' = \varepsilon$ und $u' = b'_r \ldots b'_1 \in V'^*$ für $u = b_1 \ldots b_r \in V^*$, $b_\rho \in V$, $\rho = 1, \ldots, r$, $r \in \mathbb{N}$, gesetzt. Für $u, v \in V^*$ folgt z.B. $(uv)' = v'u'$.

Für $|w| = 0$ ist, da G eine Grammatik in Greibachscher Normalform ist, die Aussage genau für $A = u$ und $m = \varepsilon$ erfüllt. Die Linksableitung $A \Longrightarrow^*_{\text{links}} u$ kann nur in 0 Schritten erfolgen, so daß sich $A = u$ und $sp(A') = A' \in \{A'\}D_n$ ergibt. Für die umgekehrte Richtung betrachten wir $sp(u') \in \{A'\}D_n$ mit $u \in V^*$. Da es in u' keine öffnende Klammern, also keine Symbole aus V gibt, kann nur $u' = A'$ und damit $u = A$ gelten.

Wir nehmen an, daß für ein Wort w_1 mit $|w_1| \geq 0$ die Behauptung erfüllt ist. Es sei $w = w_1 x$ für ein $x \in V_T$. Für den Induktionsschluß gehen wir zunächst von der linken Seite der Äquivalenz aus. Die Ableitung kann, da G in Greibachscher Normalform ist, in der Form

$$A \Longrightarrow^*_{\text{links}} w_1 B u_1 \Longrightarrow_{\text{links}} w_1 x u_2 u_1 = wu$$

mit $(B, xu_2) \in F$ und $u_2 u_1 = u$ geschrieben werden. Wegen der Induktionsannahme existiert $m_1 \in \varphi(w_1)$ mit $m_1 sp((Bu_1)') \in \{A'\}D_n$. Wegen $(B, xu_2) \in F$ und der Definition von φ folgt $B'sp(u_2) \in \varphi(x)$. Wir setzen $m = m_1 B'sp(u_2)$. Dann ergibt sich $m \in \varphi(w_1) \cdot \varphi(x) = \varphi(w)$. Zu zeigen bleibt $msp(u') \in \{A'\}D_n$. Zunächst gilt $m_1 B'sp(u'_1) = m_1 sp((Bu_1)') \in \{A'\}D_n$ sowie $sp(u_2)sp(u'_2) \in D_n$. Nach Satz 6.1(e) kann $sp(u_2)sp(u'_2)$ beliebig in das aus D_n stammende maximale Teilwort von $m_1 B'sp(u'_1)$ eingesetzt werden, um insgesamt wieder ein Wort aus $\{A'\}D_n$ zu liefern. Speziell erhalten wir

$$m_1 B'sp(u_2)sp(u'_2)sp(u'_1) = m_1 B'sp(u_2)sp(u') = msp(u') \in \{A'\}D_n.$$

Wir gehen von der rechten Seite der Äquivalenz aus. Es sei also $m \in \varphi(w)$ mit $msp(u') \in \{A'\}D_n$. Wegen $w = w_1 x$ gilt $\varphi(w) = \varphi(w_1) \cdot \varphi(x)$. Folglich existieren $m_1 \in \varphi(w_1)$, $m_2 \in \varphi(x)$ mit $m_1 m_2 = m$. Aus $m_2 \in \varphi(x)$ folgt nach der Definition von φ die Existenz einer Produktion $B \to xu_2$ aus F mit $m_2 = B'sp(u_2)$. Wir schließen

$$msp(u') = m_1 m_2 sp(u') = m_1 B'sp(u_2)sp(u'),$$

und wir wissen, daß $msp(u') \in \{A'\}D_n$ gilt. Somit existiert $u_1 \in V^*$ mit $u = u_2 u_1$ (man beachte, daß $sp(u') = sp(u'_2)sp(u'_1)$ gilt), es ist also $m_1 B'sp(u_2)sp(u'_2)sp(u'_1) \in \{A'\}D_n$.

Nach Satz 6.1(e) folgt $m_1 B' sp(u_1') = m_1 sp((Bu_1)') \in \{A'\}D_n$. Mit der Induktionsannahme und der Produktion $B \to xu_2$ erhalten wir die Ableitung

$$A \Longrightarrow^*_{\text{links}} w_1 B u_1 \Longrightarrow_{\text{links}} w_1 x u_2 u_1 = wu.$$

Damit ist der Induktionsbeweis abgeschlossen.

Es gilt jetzt

$$\begin{aligned} w \in L &\iff (X_0 \Longrightarrow^* w) \\ &\iff (X_0 \Longrightarrow^*_{\text{links}} w'X,\ (X,a) \in F,\ w'a = w) \\ &\iff (\bigvee_{m \in \varphi(w')} mX' \in \{X_0'\}D_n,\ X' \in \varphi(a),\ w'a = w) \\ &\iff (mX' \in \varphi(w) \cap \{X_0'\}D_n,\ w'a = w). \end{aligned}$$

Zum Beweis der letzten Äquivalenz überlegen wir uns, daß aus $X' \in \varphi(a)$ die Inklusion $\{X'\} \subset \varphi(a)$ und damit $\varphi(w')\{X'\} \subset \varphi(w') \cdot \varphi(a) = \varphi(w)$ folgt, also auch $mX' \in \varphi(w')\{X'\} \subset \varphi(w)$ erfüllt ist. Umgekehrt gilt nach Definition von φ die Beziehung $\varphi(a) \subset V'V^*$. Wegen $mX' \in \varphi(w) = \varphi(w') \cdot \varphi(a)$ folgt $X' \in \varphi(a)$. Wählen wir $S = X_0$, so ist der Satz bewiesen. □

Beispiel 6.1 Es sei die Grammatik

$$\begin{aligned} G = (&\{X_0, X, Y, Z, D\}, \{a, b, c, d\}, X_0, \\ &\{X_0 \to aXY, X \to bZ, Z \to dD, D \to d, Y \to c\}) \end{aligned}$$

in Greibachscher Normalform gegeben. Der Homomorphismus φ aus dem Beweis von Satz 6.2 ist durch

$$\varphi(a) = \{X_0'YX\},\ \varphi(b) = \{X'Z\},\ \varphi(c) = \{Y'\},\ \varphi(d) = \{Z'D, D'\}$$

bestimmt. Dann existiert die Linksableitung

$$X_0 \Longrightarrow_{\text{links}} aXY \Longrightarrow_{\text{links}} abZY \Longrightarrow_{\text{links}} abdDY \Longrightarrow_{\text{links}} abddY \Longrightarrow_{\text{links}} abddc.$$

Wir betrachten das Wort $abZY$, das nach zwei Ableitungsschritten entsteht. Wir setzen $w = ab$ und $u = ZY$. Mit den Bezeichnungen der oben durch Induktion bewiesenen Äquivalenz gilt $m = X_0'YXX'Z \in \varphi(w)$ und $X_0'YXX'ZZ'Y' \in \{X_0'\}D_n$. m kodiert die Ableitung von ab, es tauchen direkt oder die mit Strichen versehenen Nichtterminalzeichen der beteiligten Produktionen auf. $sp(u') = Z'Y'$ vertritt die noch abzuleitenden Symbole. Allgemein gilt, daß die letzte „öffnende Klammer“ A von m, deren Ableitung noch nicht durch die „schließende Klammer“ A' durchgeführt wurde und somit nicht in m vorkommt, dem linkesten Nichtterminalzeichen des abgeleiteten Wortes entspricht. $sp(u')$ muß also mit der entsprechenden schließenden Klammer beginnen. Da die Grammatik in Greibachscher Normalform ist, ergibt sich so ein Wort aus $\{X_0'\}D_n$.

Für die gegebene Grammatik gilt z.B.

$$\begin{aligned} \varphi(abddc) &= \{X_0'YXX'Z\}\{Z'D, D'\}\{Z'D, D'\}\{Y'\} \text{ und} \\ \varphi(abd) &= \{X_0'YXX'Z\}\{Z'D, D'\}. \end{aligned}$$

Damit erhalten wir

$$\varphi(abddc) \cap \{X_0'\}D_n \neq \emptyset,\ \varphi(abd) \cap \{X_0'\}D_n = \emptyset.$$

Nach Satz 6.2 folgt $abddc \in L(G)$, jedoch $abd \notin L(G)$. $\square$

Der nächste Satz 6.3 zeigt, daß die Aussage von Satz 6.2 schon speziell für ein Alphabet $V_2 = \{a_1, a_1', a_2, a_2'\}$ anstelle eines von der Grammatik G abhängigen Alphabets V_n gilt. Wir benötigen diesen Satz zum Beweis von Satz 6.5.

Wir geben noch eine algebraische Aussage an, die in dem Beweis von Satz 6.3 benutzt wird.

> Es sei $h : A \to B$ ein Monoidhomomorphismus. Dann ist die durch $h'(A') = \{h(a') \mid a' \in A'\}$ für alle $A' \in \mathfrak{P}'(A)$ definierte Abbildung $h' : \mathfrak{P}'(A) \to \mathfrak{P}'(B)$ ein Monoidhomomorphismus.

Für $A_1, A_2 \in \mathfrak{P}'(A)$ gilt in der Tat

$$\begin{aligned} h'(A_1A_2) &= h'(\{a_1a_2 \mid a_1 \in A_1, a_2 \in A_2\}) \\ &= \{h(a_1a_2) \mid a_1 \in A_1, a_2 \in A_2\} \\ &= \{h(a_1)h(a_2) \mid a_1 \in A_1, a_2 \in A_2\} \\ &= \{h(a_1) \mid a_1 \in A_1\}\{h(a_2) \mid a_2 \in A_2\} \\ &= h'(A_1)h'(A_2) \end{aligned}$$

und $h'(\{e_A\}) = \{h(e_A)\} = \{e_B\}$, wenn e_A und e_B die Einselemente von A bzw. B sind.

Satz 6.3 Es sei V_T ein Alphabet und $L \subset V_T^+$ eine kontextfreie Sprache. Weiter sei $V_2 = \{a_1, a_1', a_2, a_2'\}$. Dann existiert ein Monoidhomomorphismus $\varphi_2 : V_T^* \to \mathfrak{P}'(V_2^*)$, so daß die Äquivalenz

$$w \in L \iff \varphi_2(w) \cap \{a_1'\}D_2 \neq \emptyset$$

erfüllt ist.

Beweis: Das Alphabet $V = \{a_1, \ldots, a_n\}$, $n \in \mathbb{N}$, sei wie in Satz 6.2 bestimmt. Insbesondere ist $V_n = V \cup V'$. Außerdem gelte $S = a_1$. Wir definieren einen Homomorphismus $\pi' : (V \cup V')^* \to V_2^*$ durch

$$\pi'(a_i) = a_1a_2^{i-1} \text{ und } \pi'(a_i') = {a_2'}^{i-1}a_1' \text{ für } i = 1, \ldots, n.$$

Insbesondere gilt also $\pi'(S) = a_1$. Aufgrund der Definition erkennen wir sofort, daß die Äquivalenz

$$(*) \qquad w \in D_n \iff \pi'(w) \in D_2$$

erfüllt ist. Wir erweitern π' zu einem Monoidhomomorphismus

$$\pi : \mathfrak{P}'((V \cup V')^*) \to \mathfrak{P}'(V_2^*)$$

und setzen $\varphi_2 = \pi \circ \varphi$. Es folgt

$$\varphi(w) \cap \{S'\}D_n \neq \emptyset \iff \varphi_2(w) \cap \{a_1'\}D_2 \neq \emptyset.$$

Dies sehen wir wie folgt ein. Aus der linken Seite ergeben sich wegen $\pi'(D_n) \subset D_2$ die Inklusionen

$$\emptyset \neq \pi(\varphi(w) \cap \{S'\}D_n) \subset \pi\varphi(w) \cap \pi(\{S'\}D_n) \subset \varphi_2(w) \cap \{a_1'\}D_2$$

und damit die Gültigkeit der rechten Seite. Umgekehrt gelte $\varphi_2(w) \cap \{a_1'\}D_2 \neq \emptyset$. Nach der Definition von φ in dem Beweis von Satz 6.2 haben die Wörter $\varphi_2(w) = \pi(\varphi(w))$ für $w = a_1 \dots a_r$, $r \in \mathbb{N}$, die Gestalt

$$\pi'(A_1' sp(u_1) \dots A_r' sp(u_r)) \text{ für } A_i' \in V',\ u_i \in V^*,\ i = 1, \dots, r.$$

Liegt ein solches Wort auch in $\{a_1'\}D_2$, so muß nach $(*)$

$$A_1' = S' = a_1' \text{ und } sp(u_1) \dots A_r' sp(u_r) \in D_n$$

gelten, also $S' sp(u_1) \dots A_r' sp(u_r) \in \varphi(w) \cap \{S'\}D_n \neq \emptyset$. Damit ist die Äquivalenz bewiesen. Aus Satz 6.2 folgt die Behauptung. □

Wir kommen jetzt zum Satz von *Chomsky-Schützenberger.*

Satz 6.4 Es sei $L \subset V_T^*$ eine kontextfreie Sprache. Dann existiert eine reguläre Sprache R, eine Dyck-Sprache D und ein Homomorphismus μ mit

$$L = \mu(R \cap D).$$

Beweis: Ohne Beschränkung der Allgemeinheit sei $\varepsilon \notin L$. Anderenfalls zeigt der folgende Beweis die Gültigkeit von $L - \{\varepsilon\} = \mu(R \cap D)$, woraus dann $L = \mu((R \cup \{\varepsilon\}) \cap D)$ folgt.

Es sei $G = (V, V_T, X_0, F)$ eine Typ-2-Grammatik in Greibachscher Normalform mit $L(G) = L$. Wir definieren

$$Z = V \cup V_T \cup V' \cup V_T',$$

wobei V', V_T' die rechten Klammern zu den Symbolen von V und V_T bedeuten. Ein Homomorphismus $g : V_T^* \to \mathfrak{P}'(Z^*)$ wird durch

$$g(x) = \{A' sp(u) x x' \mid (A, xu) \in F\}$$

gegeben. Diese Definition entspricht bis auf xx' derjenigen von φ aus dem Beweis von Satz 6.2. Für alle $w \in V_T^+$ gilt

$$(*) \quad g(w) \cap \{X_0'\}D(V \cup V_T) \neq \emptyset \iff \varphi(w) \cap \{X_0'\}D(V) \neq \emptyset \iff w \in L.$$

Dabei ist $D(V \cup V_T)$ die Dyck-Sprache über dem Alphabet $V \cup V_T \cup V' \cup V_T'$. Die erste Äquivalenz ist erfüllt aufgrund der Definitionen von φ und g, die zweite gilt wegen Satz

6.2. Der Homomorphismus g kann wie φ in Beispiel 6.1 veranschaulicht werden, wobei zusätzlich nur die jeweiligen xx' zu berücksichtigen sind. g verliert also im Gegensatz zu φ keine Information über w.

Wir definieren den Homomorphismus $\mu : Z^* \to V_T^*$ durch

$$\mu(A) = \mu(A') = \varepsilon,\ \mu(x') = \varepsilon,\ \mu(x) = x$$

für alle $A \in V$, $x \in V_T$. Wir setzen

$$R = \{X_0\}(\bigcup_{x \in V_T} g(x))^*.$$

Da $g(x)$ endlich ist, ist R regulär. Wir behaupten

$$L = \mu(R \cap D(V \cup V_T)).$$

Zum Beweis betrachten wir ein Wort $w = x_1 \dots x_n \in L$, $x_i \in V_T$, $i = 1, \dots, n$, $n \in \mathbb{N}$. Wegen $(*)$ folgt

$$g(w) \cap \{X_0'\}D(V \cup V_T) \neq \emptyset.$$

Somit existiert ein $m \in g(w)$ mit $X_0 m \in D(V \cup V_T)$. Da $w = x_1 \dots x_n$ gilt, gibt es Wörter $m_i \in g(x_i)$ mit $m = m_1 \dots m_n$, d.h., es ist $m \in (\bigcup_{x \in V_T} g(x))^*$ und damit auch $X_0 m \in R$. Insgesamt erhalten wir also $X_0 m \in R \cap D(V \cup V_T)$. Wegen der Definition von μ und g gilt $\mu(m_i) = x_i$. Dann ergibt sich $\mu(X_0 m) = x_1 \dots x_n = w$.

Umgekehrt sei $w \in \mu(R \cap D(V \cup V_T))$. Es existiert also ein $m \in (\bigcup_{x \in V_T} g(x))^*$ mit

$$X_0 m \in D(V \cup V_T),\ \mu(X_0 m) = w.$$

Folglich gibt es eine Faktorisierung von m der Form

$$m = m_1 \dots m_n,\ m_i \in g(x_i),\ w = x_1 \dots x_n.$$

Somit gilt $m \in g(w)$. Nach Definition von g ist $m \in V'Z^*$. m beginnt also mit einer schließenden Klammer. Wegen $X_0 m \in D(V \cup V_T)$ muß diese gleich X_0' sein und damit $m \in \{X_0'\}D(V \cup V_T)$ gelten. Insgesamt erhalten wir

$$m \in g(w) \cap \{X_0'\}D(V \cup V_T) \neq \emptyset.$$

Wegen $(*)$ folgt $w \in L$. $\square$

Die Aussage des Satzes kann ähnlich Beispiel 6.1 anschaulich so interpretiert werden, daß jedes Wort aus L das homomorphe Bild der Kodierung seiner Ableitung gemäß einer Grammatik in Greibachscher Normalform ist.

Der folgende Satz von *Greibach* beweist die Existenz einer schwierigsten kontextfreien Sprache, aus der alle anderen kontextfreien Sprachen durch geeignete inverse Homomorphismen gewonnen werden können. Man kann zeigen – was wir hier jedoch nicht tun wollen – daß die Platz- und die Zeitkomplexität zur Erkennung irgendeiner kontextfreien Sprache durch die entsprechenden Komplexitäten dieser speziellen kontextfreien Sprache beschränkt sind.

Satz 6.5 Es existiert eine kontextfreie Sprache $L_{Gr} \subset V_{Gr}^*$, so daß für jede kontextfreie Sprache $L \subset V_T^+$ ein Homomorphismus $\nu : V_T^* \to V_{Gr}^*$ existiert mit

$$L = \nu^{-1}(L_{Gr}).$$

Beweis: Wir definieren

$$V_{Gr} = \{a_1, a_2, a_1', a_2', [,], +\}$$

sowie

$$L_{Gr} = \{[v_1^1 + \ldots + v_{n_1}^1][v_1^2 + \ldots + v_{n_2}^2] \ldots [v_1^l + \ldots + v_{n_l}^l] \mid n_1, \ldots, n_l \in \mathbb{N},\ l \in \mathbb{N},\\ v_j^i \in \{a_1, a_2, a_1', a_2'\}^*, \text{ es existieren } j_1, \ldots, j_l \text{ mit } v_{j_1}^1 v_{j_2}^2 \ldots v_{j_l}^l \in \{a_1'\}D_2\}.$$

Wir erkennen, daß zum Beispiel auch $[\varepsilon] = [\,]$ und $[\varepsilon + \varepsilon] = [+]$ Teilwörter von Wörtern aus L_{Gr} sind. Weiter wird der Homomorphismus $\nu : V_T^* \to V_{Gr}^*$ durch

$$\nu(x) = [u_1 + \ldots + u_n]$$

für $x \in V_T$ gegeben, wobei für den durch L bestimmten Homomorphismus φ_2 aus dem Beweis von Satz 6.3 die Gleichung $\varphi_2(x) = \{u_1, \ldots, u_n\}$ gilt. Zur eindeutigen Definition von ν nehmen wir an, daß eine beliebige Ordnung der Wörter $u_1, \ldots, u_n$ fest gewählt wird. Nun gilt

$$\begin{aligned} w' \in \nu^{-1}(L_{Gr}) \ = \ & \{w \mid w = x_1 \ldots x_l,\ \varphi_2(x_i) = \{v_1^i, \ldots, v_{n_1}^i\}, n_1, \ldots, n_l \in \mathbb{N},\ l \in \mathbb{N},\\ & \quad v_j^i \in \{a_1, a_2, a_1', a_2'\}^*, \text{ es existieren } j_1, \ldots, j_l \text{ mit } v_{j_1}^1 v_{j_2}^2 \ldots v_{j_l}^l \in \{a_1'\}D_2\}\\ \iff \ & \varphi_2(w') \cap \{a_1'\}D_2 \neq \emptyset\\ \iff \ & w' \in L. \end{aligned}$$

Die erste Äquivalenz ist aufgrund der Definitionen der beteiligten Sprachen erfüllt, die zweite ergibt sich aus Satz 6.3. Zu zeigen bleibt, daß L_{Gr} kontextfrei ist.

Wir definieren eine kontextfreie Grammatik für L_{Gr} durch Angabe ihrer Produktionen. Die Idee der Konstruktion von L_{Gr} durch die Grammatik ist, daß jedes Wort $v_{j_1}^1 \ldots v_{j_l}^l \in \{a_1'\}D_2$ mit $v_{j_i}^i \in \{a_1, a_1', a_2, a_2'\}^*$ beliebig auf $[\ldots] \ldots [\ldots]$ verstreut werden muß, wobei in jedem $[\ldots]$ ein $v_{j_i}^i$ vorkommt. Dabei ist auch $v_{j_i}^i = \varepsilon$ möglich. Im wesentlichen wird mit Hilfe der weiter unten definierten X-Produktionen ein Wort aus D_2 erzeugt, wobei mit den R-Produktionen Klammern und mit den Z-Produktionen zusätzliche „Wörter“ ε eingestreut werden. Mit Y-Produktionen werden die Teilwörter $[\ldots]$ weiter aufgefüllt.

Es sei X_0 das Anfangssymbol. Dann gibt es die X_0-Produktionen

$$X_0 \to Z[Y + a_1'X + Y]Z,$$
$$X_0 \to Z[Y + a_1'X]Z,\ X_0 \to Z[a_1'X + Y]Z,$$
$$X_0 \to Z[a_1'X]Z.$$

Dabei ist, wie wir sehen werden, X das „Anfangssymbol“ der Grammatik für die Sprache D_2. Die folgenden Produktionen erzeugen höchstens Teilwörter $[\ldots]$ von L_{Gr} mit einem $v_j^i = \varepsilon$:

$$Z \to ZZ,\ Z \to \varepsilon,\ Z \to [\,],$$
$$Z \to [Y + +Y],\ Z \to [Y+],\ Z \to [+Y].$$

Die X-Produktionen erzeugen Wörter aus D_2 mit eventuell eingestreuten Wörtern vor jedem Symbol eines Wortes aus D_2. Dabei können die Wörter aus D_2 auf verschiedene Teilwörter $[\ldots]$ verteilt werden:

$$X \to \varepsilon,\ X \to XX,$$
$$X \to Ra_1XRa_1',\ X \to Ra_2XRa_2'.$$

Die Einstreuung erfolgt durch

$$R \to \varepsilon,\ R \to]Z[Y+,$$
$$R \to]Z[,\ R \to +Y]Z[Y+,$$
$$R \to +Y]Z[.$$

Die Produktionen

$$Y \to \varepsilon,\ Y \to Yx,\ x \in \{a_1, a_2, a_1', a_2', +\}$$

dienen zur Auffüllung.

Eine genaue Betrachtung der Produktionen zeigt, daß die Sprache L_{Gr} erzeugt wird. □

Der Satz ist nur für ε-freie kontextfreie Sprachen formuliert. Gilt $\varepsilon \in L$, so folgt $L = \nu^{-1}(L_{Gr} \cup \{\varepsilon\})$.

Die hier durchgeführten Beweise der Sätze von *Chomsky-Schützenberger* und *Greibach* sind sehr algebraisch und beruhen beide auf Satz 6.2. Es besteht also eine enge Beziehung zwischen diesen beiden Sätzen. Unsere Darstellung folgt einer Arbeit von *Hotz* und *Kretschmer* [16]. In anderen Lehrbüchern (siehe z.B. [11]) werden die beiden Sätze so bewiesen, daß kein Zusammenhang zwischen ihnen erkennbar ist.

6.2 Homomorphe Charakterisierung von Typ-0-Sprachen

Satz 6.6 Es sei L eine Typ-0-Sprache. Dann existiert ein Homomorphismus h und kontextfreie Sprachen L_1 und L_2 mit

$$L = h(L_1 \cap L_2).$$

Beweis: Es sei $G = (V_N, V_T, X_0, F)$ eine Typ-0-Grammatik mit $L = L(G)$. Die Produktionenmenge bezeichnen wir mit

$$F = \{P_i \to Q_i \mid i = 1, \ldots, k\}$$

für ein geeignetes $k \in \mathbb{N}$. Wir definieren Alphabete

$$\begin{aligned} V &= V_N \cup V_T, \\ V_T' &= \{a' \mid a \in V_T\}, \\ V_{cd} &= \{c, d_1, \ldots, d_k\}, \\ V_1 &= V_N \cup V_T \cup V_T' \cup V_{cd}. \end{aligned}$$

Ist $w = a_1 \dots a_n \in V_T^*$, $a_i \in V_T$, $i = 1, \dots, n$, $n \in \mathbb{N}$, so setzen wir $w' = a_1' \dots a_n'$, und speziell gelte $\varepsilon' = \varepsilon$. Der Homomorphismus $h : V_1^* \to V_T^*$ wird durch

$$h(b) = \begin{cases} a, & \text{falls } b = a' \in V_T' \\ \varepsilon, & \text{falls } b \in V_N \cup V_T \cup V_{cd} \end{cases}$$

gegeben. In Abhängigkeit von G definieren wir zwei Sprachen über V_1, und zwar

$$\begin{aligned} L_1 &= \{R_1 P_i d_i R_2 c\, sp(R_2) sp(Q_i) sp(R_1) c \mid R_1, R_2 \in V^*,\ 1 \le i \le k\}^+ (V_T')^*, \\ L_2 &= \{X_0 d_i c \mid 1 \le i \le k\} \{sp(R_1 R_2) c R_1 d_i R_2 c \mid R_1, R_2 \in V^*,\ 1 \le i \le k\}^* \\ &\qquad \{sp(w) c w' \mid w \in V_T^*\}. \end{aligned}$$

Die Sprache L_1 hat also eine Darstellung $L_1 = L_3^+ (V_T')^*$. Dabei wird L_3 von der kontextfreien Grammatik mit den Produktionen

$$\begin{gathered} Y_0 \to Y_1 c,\ Y_1 \to x Y_1 x,\ Y_1 \to P_i d_i Y_2 sp(Q_i), \\ Y_2 \to x Y_2 x,\ Y_2 \to c \text{ für } x \in V,\ i \in \mathbb{N},\ 1 \le i \le k, \end{gathered}$$

erzeugt. Wegen Satz 2.3 und Satz 2.4 folgt $L_1 \in \mathfrak{L}_2$.

Die Sprache L_2 hat die Darstellung $L_2 = L_4 L_5^* L_6$. Dabei ist L_4 endlich und folglich kontextfrei. L_5 wird von der kontextfreien Grammatik mit den Produktionen

$$\begin{gathered} Y_0 \to Y_1 c,\ Y_1 \to x Y_1 x,\ Y_1 \to Y_2 d_i, \\ Y_2 \to x Y_2 x,\ Y_2 \to c \text{ für } x \in V,\ i \in \mathbb{N},\ 1 \le i \le k, \end{gathered}$$

erzeugt, L_6 von der kontextfreien Grammatik mit den Produktionen

$$Y_0 \to x Y_0 x',\ Y_0 \to c \text{ für } x \in V_T.$$

Satz 2.3 liefert $L_2 \in \mathfrak{L}_2$.

Zu zeigen bleibt $L = h(L_1 \cap L_2)$. Die Idee der Konstruktion beruht darauf, daß in L_1 die Produktionen von G kodiert sind. Durch die Schnittbildung kommt es zu einer Überlappung von L_3 mit L_5. Dadurch wird dann, wie wir sehen werden, eine Ableitung eines Wortes $w \in L$ erzwungen.

Wir gehen von einem Wort $w \in L$ aus. Dann existiert eine Ableitung von w gemäß G, d.h., es gibt $m \in \mathbb{N}$, Wörter $R_{j1}, R_{j2} \in V^*$ und Indizes $g(j)$ mit $1 \le g(j) \le k$, die für alle $j = 0, \dots, m$ definiert sind, so daß

$$\begin{gathered} X_0 = R_{01} P_{g(0)} R_{02} \Longrightarrow R_{01} Q_{g(0)} R_{02} = R_{11} P_{g(1)} R_{12} \Longrightarrow R_{11} Q_{g(1)} R_{12} = R_{21} P_{g(2)} R_{22} \Longrightarrow \\ \dots \Longrightarrow R_{(m-1)1} Q_{g(m-1)} R_{(m-1)2} = R_{m1} P_{g(m)} R_{m2} \Longrightarrow R_{m1} Q_{g(m)} R_{m2} = w \end{gathered}$$

eine Ableitung in G ist. Dabei gilt $R_{01} = R_{02} = \varepsilon$, $P_{g(0)} = X_0$. Für beliebige $R_1, R_2 \in V^*$ und $i \in \mathbb{N}$, $1 \le i \le k$, setzen wir

$$\begin{aligned} t(R_1, i, R_2) &= R_1 P_i d_i R_2 c\, sp(R_2) sp(Q_i) sp(R_1) c, \\ s(R_1, i, R_2) &= sp(R_1 R_2) c R_1 d_i R_2 c. \end{aligned}$$

Es gilt $t(R_1, i, R_2) \in L_3$ und $s(R_1, i, R_2) \in L_5$. Wir betrachten

$$R = t(R_{01}, g(0), R_{02})t(R_{11}, g(1), R_{12}) \dots t(R_{m1}, g(m), R_{m2})w',$$

wobei die beteiligten Daten der oben angegebenen Ableitung von w in G entnommen werden. Nach Definition von L_1 ist $R \in L_1$. Aufgrund der Ableitung von w folgt

$$\begin{aligned}
R &= \overbrace{X_0 d_{g(0)} c\, sp(R_{01} Q_{g(0)} R_{02}) c}^{t(R_{01}, g(0), R_{02})} \overbrace{R_{11} P_{g(1)} d_{g(1)} R_{12} c\, sp(R_{11} Q_{g(1)} R_{12}) c}^{t(R_{11}, g(1), R_{12})} \dots \\
&\quad \dots\dots \overbrace{sp(R_{(m-1)1} Q_{g(m-1)} R_{(m-1)2}) c}^{t(R_{(m-1)1}, g(m-1), R_{(m-1)2})} \overbrace{R_{m1} P_{g(m)} d_{g(m)} R_{m2} c\, sp(R_{m1} Q_{g(m)} R_{m2}) c}^{t(R_{m1}, g(m), R_{m2})} w' \\
&= X_0 d_{g(0)} c \underbrace{sp(R_{11} P_{g(1)} R_{12}) c R_{11} P_{g(1)} d_{g(1)} R_{12} c}_{s(R_{11} P_{g(1)}, g(1), R_{12})} \underbrace{sp(R_{21} P_{g(2)} R_{22}) c \dots\dots}_{} \\
&\quad \dots\dots \underbrace{sp(R_{m1} P_{g(m)} R_{m2}) c R_{m1} P_{g(m)} d_{g(m)} R_{m2} c}_{s(R_{m1} P_{g(m)}, g(m), R_{m2})}\, sp(w) c w' \\
&= X_0 d_{g(0)} c\, s(R_{11} P_{g(1)}, g(1), R_{12}) \dots s(R_{m1} P_{g(m)}, g(m), R_{m2}) sp(w) c w'.
\end{aligned}$$

Nach der Definition von L_2 erhalten wir also $R \in L_2$ und damit insgesamt $R \in L_1 \cap L_2$. Die Anwendung des Homomorphismus h auf R liefert $w = h(R) \in h(L_1 \cap L_2)$.

Umgekehrt gelte $w \in h(L_1 \cap L_2)$. Dann existiert ein $R \in L_1 \cap L_2$ mit $h(R) = w$. Da $R \in L_1$ gilt, kann R in der Form

$$R = t(R_{01}, g(0), R_{02})t(R_{11}, g(1), R_{12}) \dots t(R_{m1}, g(m), R_{m2})w'$$

für bestimmte $m \in \mathbb{N}$, $R_{ij} \in V^*$ und Indizes $g(i)$ mit $i = 0, \dots, m$, $j = 1, 2$ dargestellt werden. Dabei ist das Wort w' durch das gegebene w bestimmt. Da auch $R \in L_2$ gilt, kann R daneben in der Form

$$R = X_0 d_{f(0)} c\, s(\bar{R}_{11}, f(1), \bar{R}_{12}) \dots s(\bar{R}_{n1}, f(n), \bar{R}_{n2}) sp(w) c w'$$

für bestimmte $n \in \mathbb{N}$, $\bar{R}_{ij} \in V^*$ und Indizes $f(i)$ mit $i = 0, \dots, n$, $j = 1, 2$ notiert werden. Wir vergleichen diese beiden Darstellungen von R, also

$$\begin{aligned}
&\overbrace{R_{01} P_{g(0)} d_{g(0)} R_{02} c\, sp(R_{02}) sp(Q_{g(0)}) sp(R_{01}) c}\, \overbrace{R_{11} P_{g(1)} d_{g(1)} R_{12} c\, sp(R_{12}) sp(Q_{g(1)}) sp(R_{11}) c} \dots \\
&\overbrace{\dots sp(R_{(m-1)2}) sp(Q_{g(m-1)}) sp(R_{(m-1)1}) c}\, \overbrace{R_{m1} P_{g(m)} d_{g(m)} R_{m2} c\, sp(R_{m2}) sp(Q_{g(m)}) sp(R_{m1}) c}\, w' \\
&= X_0 \quad d_{f(0)} \quad \underbrace{c\, sp(\bar{R}_{11} \bar{R}_{12}) \qquad\qquad c \bar{R}_{11} \qquad d_{f(1)} \bar{R}_{12} c}\, \underbrace{sp(\bar{R}_{21} \bar{R}_{22}) \qquad\qquad c \dots}_{} \\
&\dots \underbrace{sp(\bar{R}_{n1} \bar{R}_{n2}) \qquad\qquad\qquad\qquad c \bar{R}_{n1} \qquad d_{f(n)} \bar{R}_{n2}\, c}\, sp(w) \qquad\qquad\qquad c w'.
\end{aligned}$$

Sie sind durch die Symbole c und die indizierten Symbole d gegliedert. Aufgrund der notwendig gleichen Anzahl von Symbolen c auf beiden Seiten der Gleichung folgt $n = m$. Weiterer Vergleich liefert

$g(i) = f(i)$ für $i = 0, \ldots, m$,
$R_{01} = R_{02} = \varepsilon,\ P_{g(0)} = X_0$,
$R_{i1} P_{g(i)} = \bar{R}_{i1},\ i = 1, \ldots, m$ (zwischen c und $d_{g(i)} = d_{f(i)}$),
$R_{i2} = \bar{R}_{i2},\ i = 1, \ldots, m$ (zwischen $d_{g(i)} = d_{f(i)}$ und c),
$R_{(i-1)1} Q_{g(i-1)} R_{(i-1)2} = \bar{R}_{i1} \bar{R}_{i2},\ i = 1, \ldots, m$ (zwischen zwei Symbolen c) und
$w = R_{m1} Q_{g(m)} R_{m2}$.

Aus der dritten, vierten und fünften Zeile folgt

$$R_{(i-1)1} Q_{g(i-1)} R_{(i-1)2} = \bar{R}_{i1} \bar{R}_{i2} = R_{i1} P_{g(i)} R_{i2},\ i = 1, \ldots, m.$$

Damit ist

$$X_0 = R_{01} P_{g(0)} R_{02} \Longrightarrow R_{01} Q_{g(0)} R_{02} = R_{11} P_{g(1)} R_{12} \Longrightarrow R_{11} Q_{g(1)} R_{12} = R_{21} P_{g(2)} R_{22} \Longrightarrow$$
$$\ldots \Longrightarrow R_{(m-1)1} Q_{g(m-1)} R_{(m-1)2} = R_{m1} P_{g(m)} R_{m2} \Longrightarrow R_{m1} Q_{g(m)} R_{m2} = w$$

eine Ableitung in G. Somit gilt $w \in L$. $\square$

Kapitel 7

Weitere Sprachfamilien und die Chomsky-Hierarchie

7.1 Das Wortproblem für Typ-1-Grammatiken

Liegen eine Grammatik und ein Wort vor, so ist man daran interessiert zu wissen, ob das Wort gemäß den Regeln der Grammatik aufgebaut ist. Dies Problem ist für Typ-1-Grammatiken allgemein lösbar.

Satz 7.1 Es sei G eine Typ-1-Grammatik. Dann existiert ein Algorithmus, der für ein beliebiges Wort w entscheidet, ob $w \in L(G)$ gilt oder nicht.

Beweis: Nach Definition einer Typ-1-Grammatik gilt $\varepsilon \in L(G)$ genau dann, wenn $(X_0, \varepsilon) \in F$ erfüllt ist. Dies kann durch Überprüfung der Grammatik entschieden werden.

Im folgenden gelte also $w \in V_T^+$. Wir betrachten die Folge

(1) $X_0 = P_0, P_1, \ldots, P_n = w$ mit
$n \in \mathbb{N},\ P_i \in (V_N \cup V_T)^*,\ P_i \neq P_j\ (i \neq j), |P_i| \leq |P_{i+1}|,\ i = 0, \ldots, n-1.$

Da w ein festes Wort ist, existieren nur endlich viele solcher Folgen. Wegen der Monotonie von G gilt

$$w \in L(G) \iff \text{für eine Folge (1) gilt}$$
$$(2)\quad X_0 = P_0 \Longrightarrow P_1 \Longrightarrow \ldots \Longrightarrow P_{n-1} \Longrightarrow P_n = w.$$

Man beachte dabei, daß in einer Ableitung von $w \in L(G)$ Wiederholungen der Art $P_i = P_j,\ i \neq j$, überflüssig sind, da die entsprechenden Teilableitungen entfernt werden können. Es ist immer möglich zu überprüfen, ob eine der endlichen Folgen (1) die Eigenschaft (2) erfüllt. □

Der Satz gilt auch für Typ-2- und Typ-3-Grammatiken, da Typ-3-Grammatiken Typ-2-Grammatiken sind und nach Satz 5.2 zu jeder Typ-2-Grammatik eine äquivalente Typ-1-Grammatik konstruiert werden kann. Der Satz gilt jedoch nicht für Typ-0-Grammatiken (siehe Satz 8.2).

7.2 Rekursive und rekursiv-aufzählbare Sprachen

Definition 7.1 Es sei V_T ein Alphabet.

(a) $L \subset V_T^*$ heißt *rekursiv-aufzählbar*, wenn ein Algorithmus existiert, der genau die Wörter von L, gegebenenfalls mit Wiederholungen, auflistet. Mit $\mathfrak{L}(ra)$ bezeichnen wir die *Familie der rekursiv-aufzählbaren Sprachen.*

(b) $L \subset V_T^*$ heißt *rekursiv*, wenn ein Algorithmus existiert, der für ein beliebiges Wort w entscheidet, ob $w \in L$ oder $w \notin L$ gilt. Mit $\mathfrak{L}(rek)$ bezeichnen wir die *Familie der rekursiven Sprachen.* □

Der Algorithmusbegriff sollte im Sinne der Churchschen These präzisiert werden, z.B. mit Hilfe einer Turingmaschine. Die Auflistung der Wörter kann in einem speziellen Bereich des Bandes oder in einem eigens dafür vorgesehenen Band der Turingmaschine erfolgen. Man kann annehmen, daß bei einem Übergang in einen entsprechenden speziellen Zustand die Inschrift dieses Bereiches oder Bandes das aufgelistete Wort bedeutet. Dieses darf bei der weiteren Arbeit der Turingmaschine wieder verändert werden. Bei dieser Formalisierung des Algorithmusbegriffes können wir rekursive und rekursiv-aufzählbare Sprachen wie folgt charakterisieren:

> $L \subset V_T^*$ ist genau dann eine rekursiv-aufzählbare Sprache, wenn eine Turingmaschine T existiert, die L auflistet.
> $L \subset V_T^*$ ist genau dann eine rekursive Sprache, wenn eine Turingmaschine T existiert, die bei Vorgabe beliebiger Wörter $w \in V_T^*$ hält und $L = L(T)$ liefert.

Satz 7.2 (a) Jede Typ-1-Sprache ist rekursiv.

(b) Jede Typ-0-Sprache ist rekursiv-aufzählbar.

Beweis: (a) Die Aussage ergibt sich aus Satz 7.1

(b) Es sei G eine Typ-0-Grammatik. Für jedes $k \in \mathbb{N}_0$ gibt es nur endlich viele Ableitungen in G, die mit X_0 beginnen und die die Länge k haben. Daher ist es möglich, alle Ableitungen der Länge k mit $k = 0, 1, 2, \ldots$ der Reihe nach zu überprüfen. Stammt das letzte Wort einer solchen Ableitung aus V_T^*, dann ist es in die Liste der Wörter von $L(G)$ aufzunehmen. □

Ist eine Sprache rekursiv-aufzählbar, dann existiert, wie oben ausgeführt, nach der Churchschen These eine Turingmaschine zur Auflistung von L. Diese kann in eine Turingmaschine zur Akzeptierung von L umgewandelt werden, indem zusätzlich ein Wort w vorgegeben wird. Das aufgelistete Wort wird jeweils mit w verglichen. Im Falle der Gleichheit geht die Turingmaschine in einen Endzustand über und stoppt. Nach Satz 3.11 folgt $L \in \mathfrak{L}_0$. Damit sind unter Voraussetzung der Gültigkeit der Churchschen These Typ-0-Sprachen und rekursiv-aufzählbare Sprachen äquivalent.

Satz 7.3 Es existieren rekursive Sprachen, die nicht vom Typ 1 sind.

Beweis: Der Beweis erfolgt durch ein Diagonalisierungsargument. Wir betrachten kontextsensitive Grammatiken mit Terminalalphabet $V_T = \{a, b\}$, Anfangssymbol X_0 und nichtterminale Zeichen X_i, $i \geq 0$. Beliebige Nichtterminalzeichen können stets so umbenannt werden, daß jede kontextsensitive Sprache über $\{a, b\}$ von einer solchen Grammatik erzeugt wird. Wenn

$$P_1 \to Q_1, \ldots, P_n \to Q_n$$

die Produktionen der Grammatik sind, dann wird sie vollständig durch das Wort

$$P_1 \to Q_1 \# \ldots \# P_n \to Q_n$$

festgelegt. Die in diesen Wörtern vorkommenden Symbole liegen in der Menge

$$Z = \{a, b, \to, \#\} \cup \{X_i \mid i \in \mathbb{N}_0\}.$$

Wir definieren einen Homomorphismus $\varphi : Z^* \to V_T^*$ durch

$$\begin{array}{lll} \varphi(a)=bab, & \varphi(b)=ba^2b, & \varphi(\to)=ba^3b, \\ \varphi(\#)=ba^4b, & \varphi(X_i)=ba^{i+5}b & \text{für } i \in \mathbb{N}_0 \ . \end{array}$$

Dadurch können wir die Grammatiken als Wörter über V_T verschlüsseln, zum Beispiel die Grammatik

$$X_0 \to \varepsilon, X_0 \to X_1 X_2$$

durch das Wort

$$ba^5bba^3bba^4bba^5bba^3bba^6bba^7b.$$

Umgekehrt kann man offenbar zu jedem Wort über V_T angegeben, ob es einer kontextsensitiven Grammatik entspricht oder nicht. Die Wörter über V_T werden nun aufgezählt, und zwar der Länge nach und bei gleicher Länge in lexikographischer Reihenfolge. Das leere Wort ε erhält so die Nummer 1, a die Nummer 2, b die Nummer 3, aa die Nummer 4 usw. Gleichzeitig damit erfolgt eine Numerierung der kontextsensitiven Grammatiken, und zwar entspricht die i-te Grammatik dem i-ten Wort von V_T^*, das eine kontextsensitive Grammatik definiert. Dabei kann auch $G_i = G_j$ für $i \neq j$ gelten, da eine Permutation der Produktionen eine andere Verschlüsselung liefert. Zum Beispiel ist die Grammatik mit der Nummer 1 durch die Produktion $X_0 \to \varepsilon$ gegeben und durch das Wort ba^5bba^3b mit einer Nummer k mit $2^{12} < k < 2^{13}$ verschlüsselt.

Mit Hilfe dieser Numerierung definieren wir die Sprache

$$L = \{w_i \mid w_i \notin L(G_i),\ i \in \mathbb{N}\}.$$

Es ist klar, daß L abhängig ist von der Reihenfolge der Aufzählung von V_T^*.

L ist rekursiv, denn für jedes $w \in V_T^*$ kann die Nummer i von w berechnet werden. Damit ist dann auch die kontextsensitive Grammatik G_i bestimmt. Nach Satz 7.1 existiert ein Algorithmus, der entscheidet, ob $w \in L(G_i)$ und damit $w \notin L$ gilt oder nicht.

Wenn wir annehmen, daß L kontextsensitiv ist, dann existiert ein $j \in \mathbb{N}$ mit $L = L(G_j)$. Wir betrachten speziell das Wort $w_j \in V_T^*$. Gilt $w_j \in L$, dann erhalten wir nach Definition von L die Gültigkeit von $w_j \notin L(G_j)$ und damit wegen $L = L(G_j)$ auch $w_j \notin L$, einen Widerspruch. Gilt $w_j \notin L$, dann folgt nach Definition von L die Gültigkeit von $w_j \in L(G_j)$ und damit wegen $L = L(G_j)$ auch $w_j \in L$, was ebenfalls ein Widerspruch ist. □

Satz 7.4 Es sei L eine Sprache. L ist genau dann rekursiv, wenn L und $\complement L$ rekursiv-aufzählbar sind.

Beweis: Ohne Beschränkung der Allgemeinheit können wir annehmen, daß das Komplement von L bezüglich dem Alphabet V_T von L gebildet wird.

Es sei L rekursiv. Wir bearbeiten die Wörter von V_T^* in einer festgelegten Reihenfolge (z.B. der Länge nach und bei gleicher Länge in lexikographischer Reihenfolge). Dabei wird auf jedes $w \in V_T^*$ der Algoritmus angewendet, der entscheidet, ob $w \in L$ gilt oder $w \notin L$. Im Falle $w \in L$ wird w in eine Liste für L eingetragen, im Falle $w \notin L$ in eine Liste für $\complement L$. Beide Sprachen sind also rekursiv-aufzählbar.

Umgekehrt seien L und $\complement L$ rekursiv-aufzählbar. Ist entweder L oder $\complement L$ endlich, dann sind beide Sprachen nach Satz 2.3 und Satz 1.2(d) vom Typ 3 und somit auch vom Typ 1. Nach Satz 7.2(a) folgt, daß beide Sprachen rekursiv sind. Wir nehmen nun an, daß L und $\complement L$ unendlich sind. Es existiert damit je ein Algorithmus zum Auflisten von L und von $\complement L$. Diese beiden Algorithmen werden kombiniert, indem abwechselnd die Elemente von L und $\complement L$ in eine einzige Liste eingetragen werden, etwa mit dem ersten Element von L beginnend. Jedes Element von $w \in V_T^*$ kommt in dieser Liste irgendwann einmal vor. Die Entscheidung, ob $w \in L$ oder $w \notin L$ gilt, fällt durch die Feststellung, ob das vorgelegte w an einer ungeraden oder geraden Stelle dieser Liste steht. Wird das vorgelegte w aufgelistet, wird der Algorithmus abgebrochen. Somit ist L rekursiv. □

Satz 7.5 Es existieren rekursiv-aufzählbare Sprachen, die nicht rekursiv sind.

Beweis: Der Beweis erfolgt nach Satz 7.4 durch Angabe einer rekursiv-aufzählbaren Sprache, deren Komplement nicht rekursiv-aufzählbar ist. Wir betrachten rekursiv-aufzählbare Sprachen über $V_T = \{a\}$. Offenbar existiert ein Alphabet V', so daß jeder Algorithmus, der eine Sprache über V_T auflistet, als Wort über V' ausgedrückt werden kann. Die Wörter von V'^* lassen sich als Folge $E_0, E_1, \ldots$ darstellen, die alle Algorithmen zur Auflistung von Sprachen über V_T enthält. Andere Wörter dieser Folge sind unsinnige „Algorithmen“ oder Algorithmen, die keine Sprache über V_T auflisten, die also nicht von der „richtigen Art“ sind.

Wir wollen nur kurz erwähnen, daß sich die Algorithmen der richtigen Art formal auch durch geeignete Turingmaschinen beschreiben lassen. Wählt man für Turingmaschinen eine feste Gödelisierung, so wird die Folge $0, 1, 2, \ldots$ der natürlichen Zahlen betrachtet, wobei Algorithmen der „richtigen Art“ durch solche Zahlen gegeben werden, deren Gödelnummern geeignete Turingmaschinen darstellen (für die Gödelisierung von Turingmaschinen verweisen wir auf z.B. [30]).

Wir definieren jetzt eine Sprache

$$L = \{a^n \mid n \in \mathbb{N}_0,\ a^n \text{ wird vom Algorithmus } E_n \text{ erzeugt,}\\ E_n \text{ listet eine Sprache über } V_T \text{ auf}\}.$$

Nach Satz 7.4 müssen wir zeigen, daß L rekursiv-aufzählbar ist, $\complement L$ jedoch nicht .

Wir nehmen zunächst an, daß $\complement L$ rekursiv-aufzählbar ist. Dann existiert ein $k \in \mathbb{N}_0$, so daß $\complement L$ vom Algorithmus E_k erzeugt wird. $a^k \in \complement L$ ist nach der Definition von L genau dann erfüllt, wenn a^k nicht vom Algorithmus E_k erzeugt wird. Dies ist nach der Wahl von k genau dann der Fall, wenn $a^k \notin \complement L$ gilt. Wir erhalten also

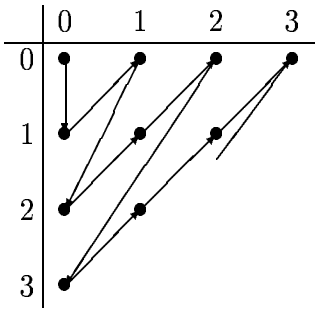
$$a^k \in \complement L \iff a^k \notin \complement L,$$

einen Widerspruch.

Zum Abschluß zeigen wir, daß L rekursiv-aufzählbar ist. Jeder Algorithmus wird auf eine solche Weise in Elementarschritte zerlegt, daß man vom i-ten Schritt ($i \in \mathbb{N}_0$) im Algorithmus E_j sprechen kann, kurz vom (i,j)-ten Schritt. Dabei soll der (i_1,j)-te Schritt nichts bewirken, falls E_j für $i_1 > i$ im i-ten Schritt endet. Diese Schritte werden in der Reihenfolge

$$(0,0),\ (1,0),\ (0,1),\ (2,0),\ (1,1),\ (0,2),\ (3,0),\ (2,1),\ \ldots$$

ausgeführt. Dies wird auch durch das Schema



dargestellt. Allgemein kann man zeigen, daß sich der k-te Schritt unserer Reihenfolge aus

$$k = \frac{1}{2} \cdot ((i+j)^2 + 3j + i) + 1$$

ergibt.

Der Algorithmus zum Auflisten der Wörter von L ergibt sich wie folgt. Falls im (i,j)-ten Schritt festgestellt wird, daß der Algorithmus E_j der „richtigen Art“ a^j erzeugt, so wird a^j in die Liste von L aufgenommen. Es werden so genau alle Wörter von L aufgelistet, L ist also rekursiv-aufzählbar. □

Da die Familie der rekursiv-aufzählbaren Sprachen nach der Bemerkung vor Satz 7.3 gleich der Familie der Typ-0-Sprachen ist, folgt aus Satz 7.4 und Satz 7.5

Satz 7.6 $\mathfrak{L}_0$ ist nicht abgeschlossen unter Komplementbildung. □

7.3 Die Chomsky-Hierarchie

Zum Abschluß geben wir den Satz an, der die Sprechweise „Chomsky-Hierarchie“ rechtfertigt.

Satz 7.7 Es gilt $\mathfrak{L}_3 \subsetneqq \mathfrak{L}_2 \subsetneqq \mathfrak{L}_1 \subsetneqq \mathfrak{L}_0$ und zusätzlich $\mathfrak{L}_1 \subsetneqq \mathfrak{L}(rek) \subsetneqq \mathfrak{L}_0 = \mathfrak{L}(ra)$.

Beweis: $\mathfrak{L}_3 \subset \mathfrak{L}_2$ und $\mathfrak{L}_1 \subset \mathfrak{L}_0$ folgen nach Definition 2.6, $\mathfrak{L}_2 \subset \mathfrak{L}_1$ ergibt sich aus Satz 5.2. $\mathfrak{L}_3 \subsetneqq \mathfrak{L}_2$ gilt, da nach Beispiel 4.2 die Sprache $\{a^n b^n \mid n \in I\!N\}$ nicht regulär ist. Weiter ist $\mathfrak{L}_2 \subsetneqq \mathfrak{L}_1$, da nach Beispiel 2.5 die Sprache $\{a^n b^n c^n \mid n \in I\!N\}$ durch eine monotone Grammatik erzeugt wird und somit nach Satz 2.7 zu $\mathfrak{L}_1$ gehört, andererseits aber nach Beispiel 5.7 keine Typ-2-Sprache ist. Schließlich gilt $\mathfrak{L}_1 \subsetneqq \mathfrak{L}(rek) \subsetneqq \mathfrak{L}_0$ nach der Churchschen These und den Sätzen 7.2, 7.3, 7.4 und 7.5. □

Die Sätze 3.4, 3.5 und 3.8 liefern außerdem

$$\mathfrak{L}_3 \subsetneqq \mathfrak{L}_{2,d} \subsetneqq \mathfrak{L}_2.$$

Kapitel 8

Entscheidbarkeitsfragen

8.1 Entscheidbarkeitsfragen für Typ-0- und Typ-1-Grammatiken

Ein *Entscheidungsproblem* ist ein Problem, das eine Ja- oder Nein-Antwort besitzt. Das Problem ist entscheidbar, wenn ein Algorithmus existiert, der für alle Fälle des Problems in endlicher Zeit die Entscheidung, also die Antwort „ja“ oder „nein“, liefert. Bei formalen Sprachen bzw. Grammatiken kann man z.B. folgende Probleme betrachten:

Problem	Frage	Fall
Leerheitsproblem	$L(G) = \emptyset$?	G
Wortproblem	$w \in L(G)$?	G, w
Äquivalenzproblem	$L(G_1) = L(G_2)$?	G_1, G_2
Unendlichkeitsproblem	$\text{card}(L(G)) = \infty$?	G

Falls ein Problem für eine Sprachfamilie $\mathfrak{L}_i$, $i = 0, 1, 2, 3$, entscheidbar ist und die Problemstellung unabhängig von $\mathfrak{L}_i$ formuliert werden kann, in der Problemstellung das spezielle i also nicht vorkommt, dann ist das Problem auch für $\mathfrak{L}_j$ mit $j > i$ entscheidbar. Ist ein Problem dagegen für eine Familie $\mathfrak{L}_i$ unentscheidbar, dann ist es auch für $\mathfrak{L}_j$ mit $j < i$ unentscheidbar.

Satz 8.1 Es sei $L \subset V_T^*$ eine Typ-0-Sprache und $a, b \notin V_T$. Dann existiert eine Typ-1-Sprache L_1 mit folgenden Eigenschaften:

(a) L_1 besteht aus Wörtern der Form $a^i bw$, $w \in L$, $i \in \mathbb{N}_0$.

(b) Zu jedem $w \in L$ existiert ein $i \in \mathbb{N}_0$ mit $a^i bw \in L_1$.

Beweis: Es sei $L = L(G)$ mit $G = (V_N, V_T, X_0, F)$. Wir definieren

$$V_N' = V_N \cup \{X_0', Y\},\ V_T' = V_T \cup \{a, b\}$$

sowie

$$F_1 = \{(P, Q) \mid (P, Q) \in F,\ |P| \leq |Q|\} \cup \{(P, Y^{|P|-|Q|}Q) \mid (P, Q) \in F,\ |P| > |Q|\},$$
$$F_2 = \{(X_0', bX_0), (Yb, ab)\} \cup \{(\alpha Y, Y\alpha) \mid \alpha \in V_N \cup V_T \cup \{b\}\}.$$

Damit erhalten wir die monotone Grammatik

$$G_1 = (V_N', V_T', X_0', F_1 \cup F_2).$$

Nach Satz 2.7 ist $L_1 = L(G_1)$ eine Typ-1-Sprache. Es wird zunächst X_0' durch bX_0 ersetzt. Dann können von X_0 aus alle Ableitungsschritte wie in G durchgeführt werden, wobei jedoch eventuell zusätzliche Symbole Y eingeführt werden. Diese Symbole können durch Produktionen $(\alpha Y, Y\alpha)$ immer nach links vor das Zeichen b geschoben werden, wo sie durch die Produktion (Yb, ab) in Zeichen a umgewandelt werden. Alle Ableitungen gemäß G können mit zusätzlichen Verschiebungen der Y simuliert werden, so daß (b) erfüllt ist. Jede Ableitung eines Terminalwortes gemäß G_1 erfolgt aber auf diese Weise, so daß auch (a) gilt. □

Satz 8.2 (Wortproblem für Typ-0-Grammatiken) Es existiert kein Algorithmus, der für eine beliebige Sprache $L \in \mathfrak{L}_0$ und ein Wort $w \in V_T^*$ entscheidet, ob $w \in L$ oder $w \notin L$ gilt.

Beweis: Nach Definition 7.1 ist $\mathfrak{L}(rek)$ die größte Sprachfamilie, für die das Wortproblem entscheidbar ist. Da es nach Satz 7.5 rekursiv-aufzählbare, also Typ-0-Sprachen gibt, die nicht rekursiv sind, folgt die Behauptung. □

Das Wortproblem für Typ-0-Grammatiken ist somit unentscheidbar. Es existieren jedoch rekursiv-aufzählbare Sprachen, nämlich die rekursiven, für die dies Problem entscheidbar ist. Außerdem ist es klar, daß es Wörter aus Sprachen von $\mathfrak{L}_0 - \mathfrak{L}(rek)$ gibt, für die speziell entschieden werden kann, ob $w \in L$ gilt oder nicht.

Satz 8.3 (Leerheitsproblem für Typ-0-Grammatiken) Es existiert kein Algorithmus, der für beliebige Typ-0-Grammatiken entscheidet, ob $L(G) = \emptyset$ gilt oder nicht.

Beweis: Wir nehmen an, daß das Leerheitsproblem entscheidbar ist. Es sei $G = (V_N, V_T, X_0, F)$ eine beliebige Grammatik und $w \in V_T^*$ ein beliebiges Wort. Wir betrachten die Grammatik

$$G_1 = (V_N \cup \{Y_0, \#\}, V_T, Y_0, F \cup \{Y_0 \to \#X_0\#, \#w\# \to \varepsilon\}).$$

Nach unserer Annahme können wir entscheiden, ob $L(G_1) = \emptyset$ gilt oder nicht. Es ist nur $L(G_1) = \emptyset$ oder $L(G_1) = \{\varepsilon\}$ möglich, wobei

$$L(G_1) = \emptyset \iff w \notin L(G)$$

gelten muß. Aus der Entscheidbarkeit des Leerheitsproblem würde also die Entscheidbarkeit des Wortproblems für Typ-0-Grammatiken folgen, was nach Satz 8.2 ein Widerspruch ist. □

Satz 8.4 Die folgenden Probleme sind für Typ-0-Grammatiken unentscheidbar:

(a) Gegeben seien zwei Typ-0-Grammatiken G_1, G_2. Gilt $L(G_1) = L(G_2)$? (Äquivalenzproblem)

(b) Es sei G_1 eine Typ-0-Grammatik. Ist $L(G_1)$ unendlich? (Unendlichkeitsproblem)
(c) Es sei G eine Typ-0-Grammatik und $w \in V_T^*$. Ist w ein Teilwort eines Wortes von $L(G)$? (Teilwortproblem für Typ-0-Grammatiken)

Beweis: Wir beginnen mit dem Äquivalenzproblem und betrachten die Typ-0-Grammatik $G_1 = (\{X\}, \{a\}, X, \{X \to X\})$ mit $L(G_1) = \emptyset$. Es sei G_2 eine beliebige andere Typ-0-Grammatik. Dann folgt

$$L(G_1) = L(G_2) \iff L(G_2) = \emptyset.$$

Aus der Entscheidbarkeit des Äquivalenzproblems würde die Entscheidbarkeit des Leerheitsproblems folgen, was im Widerspruch zu Satz 8.3 steht.

Zum Beweis von (b) geben wir eine beliebige Grammatik $G = (V_N, V_T, X_0, F)$ vor. Dazu konstruieren wir die Grammatik $G_1 = (V_N', V_T', Y_0, F')$ mit

$$V_N' = V_N \cup \{Y_0, \#\},\ V_T' = V_T \cup \{z\}$$

und

$$F' = F \cup \{Y_0 \to \#X_0\#, \#\# \to \varepsilon, \# \to z\#\} \cup \{\#a \to \# \mid a \in V_T\}.$$

Das Zeichen $\#$ kann nur durch die Produktion $\#\# \to \varepsilon$ gelöscht werden, und zwar genau dann, wenn eine Ableitung $X_0 \Longrightarrow_G^* w \in V_T^*$ existiert. Wegen der Produktion $\# \to z\#$ gilt

$$L(G) \neq \emptyset \iff L(G_1) \text{ unendlich.}$$

Da G beliebig gewählt war, wäre mit der Entscheidbarkeit des Unendlichkeitsproblem auch das Leerheitsproblem entscheidbar, was wieder im Widerspruch zu Satz 8.3 steht.

Schließlich wird (c) bewiesen. Es sei G beliebig vorgegeben. Mit einem beliebigen, aber für alle G gleichem $w \in V_T^*$ konstruieren wir eine Grammatik $G_1 = (V_N', V_T, Y_0, F')$ mit $V_N' = V_N \cup \{Y_0, \#\}$ und

$$F' = F \cup \{Y_0 \to \#X_0\#, \#\# \to w\} \cup \{\#a \to \# \mid a \in V_T\}.$$

Es folgt

$$L(G) \neq \emptyset \iff L(G_1) = \{w\}.$$

Wäre das Teilwortproblem entscheidbar, so könnten wir entscheiden, ob w ein Teilwort eines Wortes aus $L(G_1)$ ist, ob also $L(G_1) = \{w\}$ gilt oder nicht. Aufgrund der angegebenen Äquivalenz wäre dann das Leerheitsproblem entscheidbar, ein Widerspruch. □

Satz 8.5 (Leerheitsproblem für Typ-1-Grammatiken) Es existiert kein Algorithmus, der für eine beliebige Typ-1-Grammatik G entscheidet, ob $L(G) = \emptyset$ gilt oder nicht.

Beweis: Wir nehmen an, daß das Leerheitsproblem für Typ-1-Grammatiken entscheidbar ist. Es sei G eine beliebige Typ-0-Grammatik mit $L(G) = L$. Wir betrachten die Typ-1-Sprache L_1 aus Satz 8.1. Offensichtlich gilt

$$L_1 = \emptyset \iff L(G) = \emptyset.$$

Damit wäre das Leerheitsproblem für Typ-0-Grammatiken entscheidbar, was im Widerspruch zu Satz 8.3 steht. □

8.2 Das Postsche Korrespondenzproblem

Die meisten Unentscheidbarkeitsaussagen in der Theorie der formalen Sprachen lassen sich auf den Satz über die Unentscheidbarkeit des Postschen Korrespondenzproblems zurückführen. Dieser Satz wird zumeist mit Hilfe der Unentscheidbarkeit des Halteproblems bei Turingmaschinen bewiesen. Wir werden den Beweis jedoch über Satz 8.2 und damit zusammenhängend über Satz 7.5 führen, also über einen Diagonalschluß über Algorithmen.

Definition 8.1 Ein Viertupel $PCP = (V, n, \alpha, \beta)$ heißt *Postsches Korrespondenzproblem*, wenn V ein Alphabet ist, $n \in \mathbb{N}$ eine natürliche Zahl und

$$\alpha = (\alpha_1, \ldots, \alpha_n),\ \beta = (\beta_1, \ldots, \beta_n) \text{ mit } \alpha_i, \beta_i \in V^+,\ i = 1, \ldots, n,$$

zwei n-Tupel von Wörtern über V sind. Eine *Lösung für ein Postsches Korrespondenzproblem* ist eine nichtleere endliche Folge von Indizes

$$i_1, \ldots, i_k,\ 1 \leq i_1, \ldots, i_k \leq n,\ \text{mit } \alpha_{i_1} \ldots \alpha_{i_k} = \beta_{i_1} \ldots \beta_{i_k} \quad \square.$$

Beispiel 8.1 Wir betrachten das Postsche Korrespondenzproblem

$$(\{a, b\}, 3, \alpha, \beta) \text{ mit } \alpha = (a^2, b^2, ab^2),\ \beta = (a^2b, ba, b).$$

Es besitzt die Lösung 1,2,1,3, da $a^2b^2a^2ab^2 = a^2bbaa^2bb$ gilt. $\square$

Beispiel 8.2 Wir betrachten das Postsche Korrespondenzproblem

$$(\{a\}, 2, \alpha, \beta) \text{ mit } \alpha = (a^3, a^4),\ \beta = (a^2, a^3).$$

Es besitzt keine Lösung, da für jede nichtleere Indexfolge das α-Wort länger als das entsprechende β-Wort ist. $\square$

Satz 8.6 Es gibt keinen Algorithmus, der für ein beliebiges Postsches Korrespondenzproblem entscheidet, ob es eine Lösung besitzt.

Beweis: Wir nehmen an, daß es einen Algorithmus gibt, der entscheidet, ob ein beliebiges Postsches Korrespondenzproblem eine Lösung besitzt. Wir werden zeigen, daß unter dieser Voraussetzung das Wortproblem für Typ-0-Grammatiken entscheidbar ist, was ein Widerspruch zu Satz 8.2 darstellt.

Es sei $G = (V_N, V_T, a_1, F)$ eine beliebige Typ-0-Grammatik mit $V = V_N \cup V_T = \{a_1, \ldots, a_r\}$ für ein $r \in \mathbb{N}$ und $F = \{P_i \to Q_i \mid 1 \leq i \leq n\}$ für ein $n \in \mathbb{N}$. Es sei $w \in V_T^*$ beliebig gewählt. Wir konstruieren ein Postsches Korrespondenzproblem, das genau dann eine Lösung besitzt, wenn $w \in L(G)$ gilt. Aus der Entscheidbarkeit des Postschen Korrespondenzproblems würde also die Entscheidbarbeit des Wortproblems für Typ-0-Grammatiken folgen, ein Widerspruch.

Ohne Beschränkung der Allgemeinheit gelte $(a_1, a_1) \in F$. Diese Produktion kann immer zur Grammatik hinzugefügt werden, ohne die erzeugte Sprache zu verändern.

Weiter wollen wir voraussetzen, daß G ε-frei ist, also $Q_i \neq \varepsilon$ für alle i, $i = 1, \ldots, n$, gilt. Anderenfalls würden wir Produktionen $P \to \varepsilon$ durch die Produktionen $aP \to a$ und $Pa \to a$ für alle $a \in V$ ersetzen und damit eine Grammatik erhalten, die $L(G) - \{\varepsilon\}$ erzeugt (siehe auch Beweis von Satz 2.3, Seite 28). Der Beweis von Satz 8.2 beruht auf Satz 7.5 und bleibt gültig, wenn nur die Zugehörigkeit von nichtleeren Wörtern betrachtet wird. Für die im Beweis von Satz 7.5 betrachtete Sprache L gilt nämlich $\varepsilon \notin L$, d.h., für ε und L ist das Wortproblem entscheidbar. Die Nichtentscheidbarkeit des Wortproblems hängt also nicht von dem leeren Wort ab.

Wir definieren Alphabete

$$V' = \{a' \mid a \in V\} \text{ und } V_1 = V \cup V' \cup \{B, E, c, c'\},$$

wobei für $Q = a_1 \ldots a_l \in V^*$ die Definition $Q' = a_1' \ldots a_l'$ und $\varepsilon' = \varepsilon$ gegeben wird. Wir betrachten das Postsche Korrespondenzproblem

$$PCP = (V_1, 2r + 2n + 4, \alpha, \beta) \text{ mit } \alpha = (\alpha_1, \ldots, \alpha_{2r+2n+4}),\ \beta = (\beta_1, \ldots, \beta_{2r+2n+4}),$$

wobei

$$\begin{aligned}
\alpha_1 &= Ba_1c, & \beta_1 &= B, & & \\
\alpha_2 &= c', & \beta_2 &= c, & & \\
\alpha_3 &= c, & \beta_3 &= c', & & \\
\alpha_4 &= E, & \beta_4 &= c'wE, & & \\
\alpha_{4+i} &= a_i', & \beta_{4+i} &= a_i, & & 1 \leq i \leq r, \\
\alpha_{4+r+i} &= a_i, & \beta_{4+r+i} &= a_i', & & 1 \leq i \leq r, \\
\alpha_{4+2r+i} &= Q_i', & \beta_{4+2r+i} &= P_i, & & 1 \leq i \leq n, \\
\alpha_{4+2r+n+i} &= Q_i, & \beta_{4+2r+n+i} &= P_i', & & 1 \leq i \leq n,
\end{aligned}$$

gesetzt wird.

Wir zeigen zunächst, daß aus $w \in L(G)$ folgt, daß das Problem PCP eine Lösung besitzt. Da $w \in L(G)$ ist, existiert eine Ableitung von w gemäß G, d.h., es gibt $m \in \mathbb{N}$, Wörter $R_{j1}, R_{j2} \in V^*$ und Indizes $g(j)$ mit $1 \leq g(j) \leq n$, die für alle $j = 0, \ldots, m$ definiert sind, so daß

$$\begin{aligned}
a_1 = R_{01}P_{g(0)}R_{02} &\Longrightarrow R_{01}Q_{g(0)}R_{02} = R_{11}P_{g(1)}R_{12} \\
&\Longrightarrow R_{11}Q_{g(1)}R_{12} = R_{21}P_{g(2)}R_{22} \Longrightarrow \ldots \Longrightarrow R_{m1}Q_{g(m)}R_{m2} = w
\end{aligned}$$

eine Ableitung in G ist. Ohne Beschränkung der Allgemeinheit sei $m = 2m_1 - 1$ mit einem geeigneten $m_1 \in \mathbb{N}$, d.h., m ist immer ungerade. Dies können wir ggf. durch Hinzufügen eines direkten Ableitungsschritts $a_1 \Longrightarrow a_1$ erzwingen.

Wir geben eine Lösung für PCP an, also eine endliche Folge von Indizes i_j mit $1 \leq i_j \leq 2r + 2n + 4$, für die die α- und β-Wörter übereinstimmen. Für zwei Folgen T und U von Indizes ist TU die Folge von Indizes, die man erhält, wenn man U hinter T schreibt. Wir definieren zwei Abbildungen

$$\varphi, \varphi' : V^* \to \text{Menge der Folgen von Indizes}$$

durch

$$\begin{aligned}\varphi(\varepsilon) = \varphi'(\varepsilon) &= \emptyset \text{ (leere Folge)},\\ \varphi(a_{i_1}\dots a_{i_k}) &= (4+r+i_1,\dots,4+r+i_k),\\ \varphi'(a_{i_1}\dots a_{i_k}) &= (4+i_1,\dots,4+i_k) \text{ für alle } k\in\mathbb{N}.\end{aligned}$$

Es sei $U=(i_1,\dots,i_k)$, dann definieren wir

$$\alpha(U)=\alpha_{i_1}\dots\alpha_{i_k},\ \beta(U)=\beta_{i_1}\dots\beta_{i_k}.$$

Wir geben mit Hilfe der Abbildungen φ und φ' die Folgen

$$\begin{aligned}U_{2i} &= \varphi'(R_{(2i)1})(4+2r+g(2i))\varphi'(R_{(2i)2}),\\ U_{2i+1} &= \varphi(R_{(2i+1)1})(4+2r+n+g(2i+1))\varphi(R_{(2i+1)2})\end{aligned}$$

für i, $0\le i<m_1$, an. Damit sind wegen $m=2m_1-1$ die U_j für alle j, $0\le j\le m$, definiert. Für alle i, $0\le i<m_1$, gilt dann

$$\begin{aligned}\alpha(U_{2i}) &= R'_{(2i)1}Q'_{g(2i)}R'_{(2i)2},\\ \beta(U_{2i}) &= R_{(2i)1}P_{g(2i)}R_{(2i)2},\\ \alpha(U_{2i+1}) &= R_{(2i+1)1}Q_{g(2i+1)}R_{(2i+1)2},\\ \beta(U_{2i+1}) &= R'_{(2i+1)1}P'_{g(2i+1)}R'_{(2i+1)2}.\end{aligned}$$

Wir betrachten die Folge

$$U=(1)U_0(2)U_1(3)U_2(2)U_3(3)U_4(2)\dots U_{m-1}(2)U_m(4).$$

Da $m-1$ gerade ist, tritt der Wechsel zwischen (2) und (3) in U so auf, wie er notiert ist. Die Folge ist eine Lösung des Postschen Korrespondenzproblems PCP. Nach der oben angegebenen Ableitung für $w\in L(G)$ gilt nämlich

$$\alpha(U)=$$

$$\overbrace{B\quad a_1}^{\alpha_1}\ c\overbrace{R'_{01}Q'_{g(0)}R'_{02}}^{\alpha(U_0)}\ \overbrace{c'}^{\alpha_2}\overbrace{R_{11}Q_{g(1)}R_{12}}^{\alpha(U_1)}\overbrace{c}^{\alpha_3}\ \dots\ \overbrace{c'}^{\alpha_2}\overbrace{R_{m1}Q_{g(m)}R_{m2}}^{\alpha(U_m)}\overbrace{E}^{\alpha_4}=$$

$$\underbrace{B}_{\beta_1}\underbrace{R_{01}P_{g(0)}R_{02}}_{\beta(U_0)}\underbrace{c}_{\beta_2}\underbrace{R'_{11}P'_{g(1)}R'_{12}}_{\beta(U_1)}\underbrace{c'}_{\beta_3}\underbrace{R_{21}P_{g(1)}R_{22}}_{\beta(U_2)}\underbrace{c}_{\beta_2}\ \dots\ \underbrace{c'\quad w\quad E}_{\beta_4}$$

$$=\beta(U)$$

Man beachte, daß $m-1$ gerade sein muß, weil c' und nicht c in β_4 auftritt.

Umgekehrt nehmen wir an, daß das Problem PCP eine Lösung U_1 besitzt. Wir müssen zeigen, daß dann $w\in L(G)$ gilt. Es ist 1 der einzige Index j, für den α_j und β_j mit demselben Zeichen beginnen. 4 ist der einzige Index j, für den α_j und β_j mit demselben Zeichen enden. Folglich gilt

$$U_1=(1)U(4)$$

mit einer geeigneten Indexfolge U. Wir können ohne Beschränkung der Allgemeinheit annehmen, daß U weder den Index 1 noch den Index 4 enthält. Anderenfalls betrachten wir das erste Auftreten von 4 in U_1 und erhalten so

$$U_1 = (1)U_2(4)U_3(4), \text{ wobei 4 nicht in } U_2 \text{ vorkommt.}$$

Das Zeichen E tritt nur in α_4 und β_4 auf. Folglich ist $(1)U_2(4)$ eine Lösung von PCP. Wir betrachten jetzt das letzte Auftreten von 1 in $(1)U_2(4)$. Dann ergibt sich

$$(1)U_2(4) = (1)U_4(1)U_5(4), \text{ wobei 1 nicht in } U_5 \text{ vorkommt.}$$

Das Symbol B erscheint nur in α_1 und β_1, es ist also $(1)U_5(4)$ eine Lösung von PCP, und 1 und 4 treten in U_5 nicht auf. Ohne Beschränkung der Allgemeinheit kann also die obige Annahme gemacht werden.

Es sei $h : (V \cup V')^* \to V^*$ ein Homomorphismus, der durch

$$h(a') = h(a) = a \text{ für alle } a \in V$$

definiert ist. Wir beweisen die folgende Aussage (A):

> Für jede Zerlegung der Indexfolge $U = T_1T_2$ gibt es Wörter $R_1,\ R_2,\ R_3,\ R_4$ mit
>
> $$\alpha((1)T_1) = R_1R_2R_3R_4,\ \beta((1)T_1) = R_1,\ R_3 \in \{c, c'\},\ a_1 \Longrightarrow_G^* h(R_4R_2).$$

Vor dem Beweis der Aussage (A) zeigen wir, daß aus (A) die Gültigkeit von $w \in L(G)$ folgt. Es sei $T_2 = \emptyset$ die leere Folge, also $U = T_1$. Dann erhalten wir

$$\begin{aligned} &\alpha((1)U(4)) = R_1R_2R_3R_4E, \\ &\beta((1)U(4)) = R_1c'wE, \\ &a_1 \Longrightarrow_G^* h(R_4R_2). \end{aligned}$$

Da nach Voraussetzung $U_1 = (1)U(4)$ eine Lösung von PCP ist, gilt $\alpha((1)U(4)) = \beta((1)U(4))$. Wegen $R_3 \in \{c, c'\}$ muß dann

$$R_2 = \varepsilon,\ R_3 = c',\ R_4 = w,\ a_1 \Longrightarrow_G^* h(w) = w$$

folgen und damit $w \in L(G)$ gelten.

Es ist also nur noch der Beweis der Aussage (A) offen. Dieser erfolgt durch vollständige Induktion über die Länge von T_1. Für $|T_1| = 0$, d.h. $T_1 = \emptyset$, erhalten wir

$$\alpha((1)T_1) = \alpha_1 = Ba_1c,\ \beta((1)T_1) = \beta_1 = B.$$

Bei Wahl von $R_1 = B$, $R_2 = a_1$, $R_3 = c$ und $R_4 = \varepsilon$ ist die Aussage (A) erfüllt. Speziell gilt $a_1 \Longrightarrow_G^* h(a_1) = a_1$.

Wir nehmen nun an, daß (A) für $U = T_1T_2$ gilt und betrachten die Zerlegung

$$U = T_1^+T_2^+ \text{ mit } T_1^+ = T_1(q),\ 1 \leq q \leq 4 + 2r + 2n.$$

Zu zeigen ist die Existenz von Wörtern $R_1^+,\ R_2^+,\ R_3^+,\ R_4^+$ mit

$$\begin{aligned}&\alpha((1)T_1^+) = R_1^+R_2^+R_3^+R_4^+,\\ &\beta((1)T_1^+) = R_1^+,\\ &R_3^+ \in \{c, c'\},\\ &a_1 \Longrightarrow_G^* h(R_4^+R_2^+).\end{aligned}$$

Dazu betrachten wir die verschiedenen Fälle von q. Da U weder 1 noch 4 enthält, kann $q \neq 1$, $q \neq 4$ angenommen werden.

Für $q = 2$ gilt unter Benutzung der Induktionsannahme

$$\begin{aligned}\alpha((1)T_1^+) &= \alpha((1)T_1(2)) = \alpha((1)T_1)\alpha_2 = R_1R_2R_3R_4c',\\ \beta((1)T_1^+) &= \beta((1)T_1(2)) = \beta((1)T_1)\beta_2 = R_1c.\end{aligned}$$

Da $((1)U(4))$ eine Lösung von PCP ist, ist eines dieser Wörter ein Präfix des anderen Wortes. R_2 und R_4 enthalten weder c noch c'. Anderenfalls könnte $a_1 \Longrightarrow_G^* h(R_4R_2)$ nicht erfüllt sein, da c und c' nicht im Alphabet V von G vorkommen und damit $h(R_4R_2)$ nicht gebildet werden kann. Es folgt $R_2 = \varepsilon$ und $R_3 = c$. Wir wählen

$$R_1^+ = R_1c,\ R_2^+ = R_4,\ R_3^+ = c',\ R_4^+ = \varepsilon$$

und erhalten damit

$$\begin{aligned}R_1^+R_2^+R_3^+R_4^+ &= R_1cR_4c' = \alpha((1)T_1^+),\\ R_1^+ \qquad\quad &= R_1c \qquad\ = \beta((1)T_1^+),\\ R_3^+ &= c' \in \{c, c'\},\\ a_1 &\Longrightarrow_G^* h(R_4R_2) = h(R_4) = h(R_4^+R_2^+),\end{aligned}$$

wobei für den Ableitungsschritt die Induktionsannahme verwendet wird.

Für $q = 3$ gilt

$$\begin{aligned}\alpha((1)T_1^+) &= \alpha((1)T_1(3)) = \alpha((1)T_1)\alpha_3 = R_1R_2R_3R_4c,\\ \beta((1)T_1^+) &= \beta((1)T_1(3)) = \beta((1)T_1)\beta_3 = R_1c'.\end{aligned}$$

Es folgt $R_2 = \varepsilon$, $R_3 = c'$. Bei Wahl von

$$R_1^+ = R_1c',\ R_2^+ = R_4,\ R_3^+ = c,\ R_4^+ = \varepsilon$$

erhalten wir ähnlich wie zuvor das gewünschte Ergebnis.

Für $q = 4 + i$, $1 \leq i \leq r$, gilt

$$\begin{aligned}\alpha((1)T_1^+) &= \alpha((1)T_1(4+i)) = \alpha((1)T_1)\alpha_{4+i} = R_1R_2R_3R_4a_i',\\ \beta((1)T_1^+) &= \beta((1)T_1(4+i)) = \beta((1)T_1)\beta_{4+i} = R_1a_i.\end{aligned}$$

Wegen $R_3 \in \{c, c'\}$ folgt $R_2 = a_iQ$ mit einem Wort $Q \in V^*$. Wir wählen

$$R_1^+ = R_1a_i,\ R_2^+ = Q,\ R_3^+ = R_3,\ R_4^+ = R_4a_i'.$$

Neben den anderen Eigenschaften von (A) gilt dann auch

$$a_1 \Longrightarrow_G^* h(R_4R_2) = h(R_4a_iQ) = h(R_4a_i'Q) = h(R_4^+R_2^+).$$

Für $q = 4 + r + i$, $1 \leq i \leq r$, gilt

$$\begin{aligned}\alpha((1)T_1^+) &= \alpha((1)T_1)\alpha_{4+r+i} = R_1R_2R_3R_4a_i,\\ \beta((1)T_1^+) &= \beta((1)T_1)\beta_{4+r+i} = R_1a_i'.\end{aligned}$$

Es folgt $R_2 = a_i'Q$ mit einem $Q \in V^*$. Bei der Wahl von

$$R_1^+ = R_1a_i',\ R_2^+ = Q,\ R_3^+ = R_3,\ R_4^+ = R_4a_i$$

erhalten wir das gewünschte Resultat, insbesondere auch

$$a_1 \Longrightarrow_G^* h(R_4R_2) = h(R_4a_i'Q) = h(R_4a_iQ) = h(R_4^+R_2^+).$$

Für $q = 4 + 2r + i$, $1 \leq i \leq n$, gilt

$$\begin{aligned}\alpha((1)T_1^+) &= \alpha((1)T_1)\alpha_{4+2r+i} = R_1R_2R_3R_4Q_i',\\ \beta((1)T_1^+) &= \beta((1)T_1)\beta_{4+2r+i} = R_1P_i.\end{aligned}$$

Es folgt $R_2 = P_iQ$ mit einem $Q \in V^*$. Wir wählen

$$R_1^+ = R_1P_i,\ R_2^+ = Q,\ R_3^+ = R_3,\ R_4^+ = R_4Q_i'.$$

Es gilt

$$a_1 \Longrightarrow_G^* h(R_4R_2) = h(R_4P_iQ),\ h(Q_i') = Q_i,\ (P_i, Q_i) \in F.$$

Damit erhalten wir

$$a_1 \Longrightarrow_G^* h(R_4P_iQ) = h(R_4)P_ih(Q) \Longrightarrow h(R_4)Q_ih(Q) = h(R_4Q_i'Q) = h(R_4^+R_2^+).$$

Für $q = 4 + 2r + n + i$, $1 \leq i \leq n$, gilt schließlich

$$\begin{aligned}\alpha((1)T_1^+) &= \alpha((1)T_1)\alpha_{4+2r+n+i} = R_1R_2R_3R_4Q_i,\\ \beta((1)T_1^+) &= \beta((1)T_1)\beta_{4+2r+n+i} = R_1P_i'.\end{aligned}$$

Es folgt $R_2 = P_i'Q$ mit einem $Q \in V^*$. Wir wählen

$$R_1^+ = R_1P_i',\ R_2^+ = Q,\ R_3^+ = R_3,\ R_4^+ = R_4Q_i.$$

Der weitere Beweis ergibt sich wie im vorhergehenden Fall, jedoch mit $h(P_i') = P_i$ anstelle von $h(Q_i') = Q_i$.

Alle möglichen Werte von q sind damit betrachtet. Die Aussage (A) ist also erfüllt. □

Eine stärkere Version von Satz 8.6 liefert

Satz 8.7 Es sei V_2 ein Alphabet mit $\text{card}(V_2) = 2$. Dann existiert kein Algorithmus, der zu einem beliebigen Postschen Korrespondenzproblem

$$PCP = (V_2, n, \alpha, \beta),\ \alpha = (\alpha_1, \ldots, \alpha_n),\ \beta = (\beta_1, \ldots, \beta_n)$$

entscheidet, ob es eine Lösung besitzt oder nicht.

Beweis: Es sei $V_2 = \{a, b\}$. Wir nehmen das Gegenteil der Aussage des Satzes an. Es sei

$$PCP' = (V, n, \alpha', \beta') \text{ mit } V = \{a_1, \ldots, a_r\}$$

ein beliebiges Postsches Korrespondenzproblem. Wir müssen zeigen, daß im Widerspruch zu Satz 8.6 entschieden werden kann, ob PCP' eine Lösung besitzt.

Wir definieren einen Homomorphismus $h : V^* \to V_2^*$ durch

$$h(a_i) = ba^ib,\ 1 \leq i \leq r.$$

Dann seien α und β die n-Tupel von Wörtern über $\{a, b\}$, die man aus α' und β' durch homomorphe Ersetzung der einzelnen Wörter gemäß h erhält. Damit ist dann ein Postsches Korrespondenzproblem $PCP = (V_2, n, \alpha, \beta)$ definiert. Offensichtlich gilt

$$PCP \text{ besitzt eine Lösung} \iff PCP' \text{ besitzt eine Lösung.}$$

Da nach unserer Annahme PCP eine Lösung hat, hat auch PCP' eine. □

8.3 Unentscheidbarkeitsaussagen für kontextfreie Grammatiken

Bevor wir Unentscheidbarkeitsaussagen für kontextfreie Grammatiken treffen können, benötigen wir noch eine Definition sowie einen Satz.

Definition 8.2 Es sei $V_2 = \{a, b\}$, $V_3 = \{a, b, c\}$. Dann definieren wir die Sprache

$$L_{sp} = \{w_1 c w_2 c\, sp(w_2) c\, sp(w_1) \mid w_1, w_2 \in V_2^*\}$$

über V_3. Weiter sei ein Postsches Korrespondenzproblem $PCP = (V_2, n, \alpha, \beta)$ gemäß Satz 8.7 gegeben. Dann setzen wir

$$\begin{aligned} L(\alpha) &= \{ba^{i_k}ba^{i_{k-1}}b \ldots ba^{i_1}c\alpha_{i_1} \ldots \alpha_{i_k} \mid k \in \mathbb{N},\ 1 \leq i_j \leq n\}, \\ L(\beta) &= \{ba^{i_k}ba^{i_{k-1}}b \ldots ba^{i_1}c\beta_{i_1} \ldots \beta_{i_k} \mid k \in \mathbb{N},\ 1 \leq i_j \leq n\}, \\ L(PCP) &= L(\alpha)\{c\}sp(L(\beta)). \quad \square \end{aligned}$$

Die Sprache $L(PCP)$ kann auch durch

$$\begin{aligned} &L(PCP) \\ &= \{ba^{i_k}b \ldots ba^{i_1}c\alpha_{i_1} \ldots \alpha_{i_k}c\, sp(\beta_{j_l}) \ldots sp(\beta_{j_1})ca^{j_1}b \ldots ba^{j_l}b \mid k, l \in \mathbb{N},\ 1 \leq i_r, j_s \leq n\} \end{aligned}$$

dargestellt werden.

Satz 8.8 L_{sp}, $V_3^* - L_{sp}$ sowie die Sprachen $L(PCP)$ und $V_3^* - L(PCP)$ für alle Postschen Korrespondenzprobleme gemäß Satz 8.7 sind kontextfrei.

Beweis: L_{sp} wird durch die kontextfreie Grammatik mit den Produktionen

$$X_0 \to yX_0y,\ X_0 \to cX_1c,\ X_1 \to yX_1y,\ X_1 \to c \text{ für alle } y \in V_2$$

erzeugt. Die kontextfreie Grammatik mit den Produktionen

$$X_0 \to ba^jX_0\alpha_j,\ X_0 \to ba^jc\alpha_j \text{ für alle } j \text{ mit } 1 \leq j \leq n$$

erzeugt $L(\alpha)$, entsprechend ist $L(\beta)$ eine kontextfreie Sprache. Da das Spiegelbild einer kontextfreien Sprache wieder kontextfrei ist (offensichtlich ist allgemein das Spiegelbild einer Typ-i-Sprache wieder vom Typ i), erhalten wir mit Hilfe von Satz 2.3 die Gültigkeit von $L(PCP) \in \mathfrak{L}_2$.

Wir zeigen nun, daß $V_3^* - L_{sp}$ kontextfrei ist. Diese Sprache läßt sich darstellen als

$$V_3^* - L_{sp} = L_1 \cup L_2 \cup L_3,$$

wobei

(1) L_1 aus allen Wörtern $w \in V_3^*$ besteht, für die die Anzahl der Symbole c in w verschieden von 3 ist,

(2) L_2 aus allen Wörtern über V_3 der Form

$$w_1cw_2cw_3cw_4,\ w_1, w_2, w_3, w_4 \in V_2^*,\ w_4 \neq sp(w_1)$$

und

(3) L_3 aus allen Wörtern über V_3 der Form

$$w_1cw_2cw_3cw_4,\ w_1, w_2, w_3, w_4 \in V_2^*,\ w_3 \neq sp(w_2)$$

besteht.

Diese Sprachen sind kontextfrei. Wir können L_1 schreiben als

$$L_1 = \{a,b,c\}^*(\{c\}\{a,b,c\}^*)^4 \cup \{a,b\}^*(\{c\}\{a,b\}^*)^2 \cup \{a,b\}^*\{c\}\{a,b\}^* \cup \{a,b\}^*.$$

In der ersten Menge der Vereinigung befinden sich alle Wörter über V_3 mit mindestens 4 Vorkommen von c, in der zweiten solche mit genau 2, in der dritten mit genau einem und in der letzten mit keinem Vorkommen von c. Die Sprache L_1 läßt sich also durch einen regulären Ausdruck darstellen und ist somit erst recht vom Typ 2.

Wir betrachten weiter die Typ-3-Grammatik $G_1 = (\{X_1\}, V_3, X_1, F_1)$ mit $F_1 = \{X_1 \to \varepsilon, X_1 \to X_1a, X_1 \to X_1b\}$, die $L(G_1) = \{a,b\}^*$ erzeugt. Dazu konstruieren wir die kontextfreie Grammatik $G_2 = (\{X_1, X_2\}, V_3, X_2, F_2)$ mit

$$\begin{aligned} F_2 = F_1 \cup \{&X_2 \to aX_2a,\ X_2 \to bX_2b, X_2 \to a(X_1c)^3, X_2 \to a(X_1c)^3X_1b,\\ &X_2 \to b(X_1c)^3, X_2 \to b(X_1c)^3X_1a, X_2 \to (cX_1)^3a, X_2 \to (cX_1)^3b\}. \end{aligned}$$

Eine Ableitung vermöge G_2 beginnt mit der Teilableitung

$$X_2 \Longrightarrow^* wX_2sp(w),\ w \in V_2^*.$$

Um ein Wort über V_3 zu erhalten, muß eine Produktion aus $F_2 - F_1$ mit dem Zeichen X_1 auf der rechten Seite angewendet werden. Diese Produktionen sichern $w_4 \neq sp(w_1)$. Insgesamt gilt offenbar $L_2 = L(G_2)$, d.h., L_2 ist kontextfrei.

Wir definieren die Grammatik $\bar{G} = (\{X_1, X_3\}, V_3, X_3, F_3)$ mit

$$\begin{aligned} F_3 = F_1 \cup \{&X_3 \to aX_3a,\ X_3 \to bX_3b, X_3 \to aX_1c, X_3 \to bX_1c,\\ &X_3 \to aX_1cX_1b, X_3 \to bX_1cX_1a, X_3 \to cX_1a, X_3 \to cX_1b\}. \end{aligned}$$

Wir erkennen, daß $\bar{L} = L(\bar{G})$ aus allen Wörtern der Form

$$w_2cw_3,\ w_2, w_3 \in \{a,b\}^*,\ w_3 \neq sp(w_2),$$

besteht. Da $\bar{L}$ kontextfrei ist, ist auch $L_3 = \{a,b\}^*\{c\}\bar{L}\{c\}\{a,b\}^*$ und damit $V_3^* - L_{sp}$ kontextfrei.

Wir kommen schließlich zum Beweis der Kontextfreiheit von $V_3^* - L(PCP)$. Wir werden zeigen, daß

$$V_3^* - L(PCP) = \bigcup_{i=1}^{6} L_i \text{ mit } L_i \in \mathfrak{L}_2,\ i = 1, \ldots, 6,$$

gilt. Wir wählen

$$L_1 = \{a,b\}^* \cup \{a,b\}^*\{c\}\{a,b\}^* \cup (\{a,b\}^*\{c\})^2\{a,b\}^* \cup (\{a,b\}^*\{c\})^4V_3^*.$$

Diese Sprache enthält alle Wörter aus V_3^* mit mehr oder weniger als 3 Vorkommen von c. Sie ist ersichtlich regulär und damit auch kontextfrei. Weiter setzen wir

$$L_2 = \{a,c\}V_3^* \cup V_3^*\{a,c\} \cup V_3^*\{c^2\}V_3^*.$$

Dies sind alle Wörter aus V_3^*, die mit a oder c beginnen oder enden oder c^2 als Teilwort besitzen. Auch L_2 ist regulär und kontextfrei. Die Sprache L_3 wird durch

$$L_3 = \{a,b\}^*\{b^2\}\{a,b\}^*\{c\}V_3^* \cup V_3^*\{c\}\{a,b\}^*\{b^2\}\{a,b\}^* \cup \{a,b\}^*\{bc\}V_3^* \cup V_3^*\{cb\}\{a,b\}^*$$

gegeben. Sie besteht aus allen Wörtern, bei denen das Teilwort b^2 vor dem ersten c oder hinter dem letzten c steht oder b sich unmittelbar links vom ersten bzw. unmittelbar rechts vom letzten Vorkommen von c befindet. Auch diese Sprache ist regulär und kontextfrei. Als letzte reguläre und kontextfreie Sprache definieren wir

$$L_4 = \{a,b\}^*\{a^{n+1}\}\{a,b\}^*\{c\}V_3^* \cup V_3^*\{c\}\{a,b\}^*\{a^{n+1}\}\{a,b\}^*.$$

L_4 besteht aus allen Wörtern, die ein Teilwort a^h, $h \geq n+1$, vor dem ersten oder hinter dem letzten Vorkommen von c haben.

Wir setzen

$$H = V_3^* - \bigcup_{i=1}^{4} L_i,$$

d.h., nach den Definitionen der Sprachen L_1 bis L_4 besteht H aus allen Wörtern der Form

$$ba^{i_k} \dots ba^{i_1} cucvca^{j_1} b \dots a^{j_l} b \text{ mit } u, v \in \{a,b\}^+,\ k, l \in \mathbb{N},\ 1 \leq i_r, j_s \leq n.$$

Offensichtlich gilt $L(PCP) \subsetneqq H$ und damit

$$V_3^* - H = \bigcup_{i=1}^{4} L_i \subsetneqq V_3^* - L(PCP).$$

Durch H ist also zuviel abgezogen worden, und zwar die Wörter der obigen Gestalt mit $u \neq \alpha_{i_1} \dots \alpha_{i_k}$ oder $sp(v) \neq \beta_{j_1} \dots \beta_{j_l}$. Diese fehlenden Elemente werden durch L_5 und L_6 gegeben.

Wir betrachten $\alpha = (\alpha_1, \dots, \alpha_n)$ aus dem Postschen Korrespondenzproblem PCP. Für $i = 1, \dots, n$ definieren wir die endlichen Mengen

$$D(\alpha_i) = \{u \mid u \in \{a,b\}^+,\ |u| < |\alpha_i|\},$$
$$J(\alpha_i) = \{u \mid u \in \{a,b\}^+,\ |u| = |\alpha_i|,\ u \neq \alpha_i\}.$$

Damit erhalten wir die reguläre Sprache

$$\begin{aligned} M(\alpha) = {} & \{c\}\{a,b\}^+ \cup \{b\}\{a,b\}^*\{c\} \\ & \cup \bigcup_{i=1}^{n} \bigcup_{u \in D(\alpha_i)} (\{ba^i cu\} \cup \{ba^i b\}\{a,b\}^*\{cu\}) \\ & \cup \bigcup_{i=1}^{n} \bigcup_{u \in J(\alpha_i)} (\{ba^i c\}\{a,b\}^*\{u\} \cup \{ba^i b\}\{a,b\}^*\{c\}\{a,b\}^*\{u\}). \end{aligned}$$

In den beiden letzten Zeilen steht am Ende eines zum Index i gehörenden Wortes nie das jeweilige α_i. Auf dieselbe Weise kann die reguläre Sprache $M(\beta)$ aus dem $\beta = (\beta_1, \dots, \beta_n)$ des Postschen Korrespondenzproblems PCP konstruiert werden.

Wir nehmen an, daß $M(\alpha)$ von der kontextfreien Grammatik $G = (V_N, V_3, X_0, F)$ erzeugt wird. Dann bilden wir die kontextfreie Grammatik $G' = (V_N', V_3, X_0', F')$ mit

$$V_N' = V_N \cup \{X_0'\},\ F' = F \cup \{X_0' \to X_0, X_0' \to ba^i X_0' \alpha_i \mid 1 \leq i \leq n\}.$$

Die Sprache $L(G')$ besitzt die folgenden Eigenschaften:

- Durch Betrachtung von $M(\alpha)$ erkennen wir, daß jedes Wort aus $L(G')$ genau ein Zeichen c enthält.
- Es gilt $L(G') \cap L(\alpha) = \emptyset$, da die Wörter von $L(G')$ außen zwar wie die von $L(\alpha)$ aufgebaut sind, die Teilwörter in der Mitte jedoch aus $M(\alpha)$ stammen. Damit ist insgesamt die Gestalt von Wörtern aus $L(\alpha)$, also Wörter der Form $ba^{i_k} \dots ba^{i_1} cw$ mit $w = \alpha_{i_1} \dots \alpha_{i_k}$, nicht erfüllt.

- Wegen der Gestalt von $M(\alpha)$ und F' gilt $\{ba^{i_k} \dots ba^{i_1}cw \mid w \neq \alpha_{i_1} \dots \alpha_{i_k},\ w \in \{a,b\}^*\} \subset L(G')$.

Wir setzen

$$L_5 = L(G')\{c\}V_3^*.$$

Diese Sprache ist kontextfrei. Wegen $L(G') \cap L(\alpha) = \emptyset$ gilt auch $L_5 \cap L(PCP) = \emptyset$ und damit

$$L_5 \subset V_3^* - L(PCP).$$

Werden Wörter aus L_5 zu $V_3^* - H$ hinzugefügt, so werden die Wörter aus $L_5 \cap H$ echt hinzugefügt, während die Wörter aus $L_5 \cap (V_3^* - H)$ schon zuvor durch $\bigcup_{i=1}^{4} L_i$ berücksichtigt worden sind. $L_5 \cap H$ besteht aus allen Wörtern

$$ba^{i_k} \dots ba^{i_1}cucvca^{j_1}b \dots a^{j_l}b \text{ mit } u,v \in \{a,b\}^+,\ u \neq \alpha_{i_1} \dots \alpha_{i_k},\ k,l \in \mathbb{N},\ 1 \leq i_r, j_s \leq n.$$

Wird die kontextfreie Sprache $sp(M(\beta))$ durch die kontextfreie Grammatik $G_1 = (V_{1N}, V_3, Y_0, F_1)$ erzeugt, so konstruieren wir die Grammatik $G_1' = (V_{1N}', V_3, Y_0', F_1')$ mit

$$V_{1N}' = V_{1N} \cup \{Y_0'\},\ F_1' = F_1 \cup \{Y_0' \to Y_0, Y_0' \to sp(\beta_i)Y_0'a^ib \mid 1 \leq i \leq n\}.$$

Wir setzen

$$L_6 = V_3^*\{c\}L(G_1').$$

Analog den vorhergehenden Überlegungen stellen wir fest, daß $L_6 \cap H$ aus allen Wörtern

$$ba^{i_k} \dots ba^{i_1}cucvca^{j_1}b \dots a^{j_l}b \text{ mit } u,v \in \{a,b\}^+,\ sp(v) \neq \beta_{j_1} \dots \beta_{j_l},\ k,l \in \mathbb{N},\ 1 \leq i_r, j_s \leq n,$$

besteht. Insgesamt erhalten wir

$$(V_3^* - H) \cup L_5 \cup L_6 = \bigcup_{i=1}^{6} L_i = V_3^* - L(PCP).$$

$V_3^* - L(PCP)$ ist also eine kontextfreie Sprache. □

Satz 8.9 Jedes der folgenden Probleme ist unentscheidbar für kontextfreie Grammatiken G_1 und G_2 mit $L_1 = L(G_1)$, $L_2 = L(G_2)$:

(a)	Ist $L_1 \cap L_2 = \emptyset$?	(b)	Ist $L_1 \cap L_2$ unendlich?
(c)	Ist $L_1 \cap L_2$ regulär?	(d)	Ist $L_1 \cap L_2$ kontextfrei?
(e)	Ist $\complement L_1 = \emptyset$?	(f)	Ist $\complement L_1$ unendlich?
(g)	Ist $\complement L_1$ regulär?	(h)	Ist $\complement L_1$ kontextfrei?
(i)	Ist L_1 regulär?	(j)	Gilt $L_1 \subset L_2$?
(k)	Gilt $L_1 = L_2$?	(l)	Ist G_1 eindeutig?

Beweis: Wir zeigen, daß sich aus jedem Algorithmus zur Entscheidung eines der obigen Probleme ein Algorithmus zur Entscheidung des speziellen Postschen Korrespondenzproblems aus Satz 8.7 ergibt, was einen Widerspruch zur Unentscheidbarkeit dieses Problems darstellt.

Wir betrachten ein beliebiges Postsches Korrespondenzproblem PCP gemäß Satz 8.7. Es seien $L(PCP)$ und L_{sp} die kontextfreien Sprachen aus Satz 8.8. Wir bilden die Sprache $L' = L(PCP) \cap L_{sp}$. Sie besteht aus allen Wörtern der Form

$$(1) \qquad \begin{gathered} ba^{i_k} \ldots ba^{i_1} c\alpha_{i_1} \ldots \alpha_{i_k} c\, sp(\beta_{i_k}) \ldots sp(\beta_{i_1}) ca^{i_1} b \ldots a^{i_k} b \\ \text{mit } k \in \mathbb{N},\ 1 \leq i_j \leq n,\ \alpha_{i_1} \ldots \alpha_{i_k} = \beta_{i_1} \ldots \beta_{i_k}. \end{gathered}$$

Wir erkennen sofort, daß

$$L(PCP) \cap L_{sp} \neq \emptyset \iff PCP \text{ besitzt eine Lösung}$$

gilt. Aus der Entscheidbarkeit des Problems (a) würde die Entscheidbarkeit des speziellen Postschen Korrespondenzproblems folgen, ein Widerspruch.

Ist $(i_1, \ldots, i_k)$ eine Lösung von PCP, so ist $(i_1, \ldots, i_k, i_1, \ldots, i_k)$ ebenfalls eine Lösung von PCP. Dieses Argument kann iteriert werden, so daß

$$L(PCP) \cap L_{sp} \neq \emptyset \iff L(PCP) \cap L_{sp} \text{ unendlich}$$

folgt. Damit ist (b) aufgrund von (a) unentscheidbar.

Zum Beweis der Unentscheidbarkeit der Probleme (c) und (d) zeigen wir zunächst, daß unabhängig von der Wahl des Postschen Korrespondenzproblems PCP die Sprache $L(PCP) \cap L_{sp}$ keine unendliche kontextfreie Sprache L enthält. Wir nehmen das Gegenteil für ein geeignetes Problem PCP an. Dann kann nach Satz 5.9 jedes hinreichend lange Wort von L in der Form

$$(2) \qquad u_1 v_1 v v_2 u_2 \text{ mit } v_1 v_2 \neq \varepsilon,\ u_1 v_1^i v v_2^i u_2 \in L \text{ für alle } i \in \mathbb{N}_0$$

geschrieben werden. Ein solches Wort von $L \subset L'$ hat auch die Form (1), d.h.

$$w_1 c w_2 c w_3 c w_4 = u_1 v_1 v v_2 u_2 \text{ mit } w_1, w_2, w_3, w_4 \in V_2^*.$$

Wäre c ein Teilwort von $v_1 v_2$, so würde $u_1 v_1^4 v v_2^4 u_2 \in L$ mindestens 4 Vorkommen von c enthalten, ein Widerspruch. Es folgt, daß $u_1 v_1^2 v v_2^2 u_2 \in L$ aus $u_1 v_1 v v_2 u_2$ gewonnen wird, indem in mindestens einem und höchstens zwei der Wörter w_j, $1 \leq j \leq 4$, iteriert wird. Dann kann aber aus dem Wort $u_1 v_1 v v_2 u_2$ der Form (1) durch die Iterierung von $v_1 v_2$ kein Wort der Form (1) entstehen, es ist also $u_1 v_1^2 v v_2^2 u_2 \notin L$, ein Widerspruch. Somit enthält $L' = L(PCP) \cap L_{sp}$ keine unendliche kontextfreie Sprache. Nach dem Beweis von (b) ist $L' \neq \emptyset$ und endlich nicht möglich. Folglich ist $L' = \emptyset$, oder L' ist eine unendliche und nicht kontextfreie Sprache. Somit ist L' genau dann kontextfrei oder regulär, wenn $L' = \emptyset$ gilt. Wegen

$$L' = L(PCP) \cap L_{sp} = \emptyset \iff PCP \text{ besitzt keine Lösung}$$

sind daher die Probleme (c) und (d) unentscheidbar.

Zur Unentscheidbarkeit von (e) bis (h) betrachten wir die Sprache

$$L'' = V_3^* - (L(PCP) \cap L_{sp}) = (V_3^* - L(PCP)) \cup (V_3^* - L_{sp}),$$

die nach Satz 8.8 kontextfrei ist. Es ist

$$\complement L'' = L(PCP) \cap L_{sp} = L'.$$

Aus der Unentscheidbarkeit von (a) bis (d) folgt die Unentscheidbarkeit der Probleme (e) bis (h).

Wir kommen zum Problem (i). Wir haben eben gezeigt, daß $L' = L(PCP) \cap L_{sp} = \emptyset$ gilt oder L' unendlich und nicht kontextfrei, also erst recht nicht regulär, ist. Unter Verwendung von Satz 1.2(d) erhalten wir

$$\begin{aligned} L'' \text{ regulär} &\iff \complement L'' \text{ regulär} \iff L(PCP) \cap L_{sp} = \emptyset \\ &\iff PCP \text{ besitzt keine Lösung.} \end{aligned}$$

Damit folgt die Unentscheidbarkeit von (i) aus Satz 8.7.

Wäre (k) entscheidbar, so würde ein Algorithmus existieren, der zu entscheiden erlaubt, ob $L'' = V_3^*$ gilt. Wegen

$$L'' = V_3^* \iff L(PCP) \cap L_{sp} = \emptyset \iff PCP \text{ besitzt keine Lösung}$$

steht dies im Widerspruch zu Satz 8.7.

Daraus folgt sofort die Unentscheidbarkeit des Problems (j). Anderenfalls ließe sich nämlich entscheiden, ob $L(G_1) \subset L(G_2)$ und $L(G_2) \subset L(G_1)$ und damit $L(G_1) = L(G_2)$ gilt.

Wir betrachten zum Abschluß das Problem (l). Mit einem Postschen Korrespondenzproblem PCP gemäß Satz 8.7 betrachten wir die kontextfreie Grammatik $G(PCP)$ mit Produktionen

$$\begin{aligned} &X_0 \to X_\alpha, \quad && X_0 \to X_\beta, \\ &X_\alpha \to ba^i c\alpha_i, \quad && X_\alpha \to ba^i X_\alpha \alpha_i, \quad i = 1, \ldots, n, \\ &X_\beta \to ba^i c\beta_i, \quad && X_\beta \to ba^i X_\beta \beta_i, \quad i = 1, \ldots, n. \end{aligned}$$

X_0 ist das Anfangssymbol. Wir erkennen sofort, daß $L(G(PCP)) = L(\alpha) \cup L(\beta)$ ist. Nach Definition 5.3 ist $G(PCP)$ mehrdeutig, wenn ein $w \in L(G(PCP))$ mit mindestens 2 verschiedenen Linksableitungen $X_0 \Longrightarrow^*_{\text{links}} w$ existiert. Die Wörter von $L(\alpha)$ können aus X_α jeweils nur auf genau eine Weise abgeleitet werden. Das gleiche gilt für $L(\beta)$ mit dem Zeichen X_β. Folglich sind für ein Wort aus $L(G(PCP))$ höchstens zwei Linksableitungen möglich, und zwar über X_α oder X_β. Es gilt nun

$$\begin{aligned} G(PCP) \text{ mehrdeutig} &\iff \bigvee_{w \in L(G(PCP))} w = ba^{i_k} \ldots ba^{i_1} c\alpha_{i_1} \ldots \alpha_{i_k} \\ &\qquad\qquad = ba^{i_k} \ldots ba^{i_1} c\beta_{i_1} \ldots \beta_{i_k} \\ &\iff \alpha_{i_1} \ldots \alpha_{i_k} = \beta_{i_1} \ldots \beta_{i_k} \\ &\iff PCP \text{ besitzt eine Lösung} \end{aligned}$$

Damit ist das Problem (l) unentscheidbar. □

8.4 Zusammenfassung von Entscheidbarkeits- und Unentscheidbarkeitsaussagen

Wir fassen die Entscheidbarkeits- und Unentscheidbarkeitsaussagen in einer Tabelle zusammen. Der Buchstabe E bedeutet, daß das entsprechende Problem entscheidbar ist, U steht für die Unentscheidbarkeit, T für eine immer wahre Aussage und - ist für die Fälle notiert, bei denen die entsprechende Frage nicht anwendbar ist. In der rechten Spalte stehen die jeweiligen Begründungen, wobei z.B. beim Wortproblem der Eintrag 7.1|8.2 bedeutet, daß die Entscheidbarkeit für den Typ 1 durch Satz 7.1 und die Unentscheidbarkeit für den Typ 0 durch Satz 8.2 gegeben wird. Die Entscheidbarkeit für die Typen 3 und 2 folgt dann nach der Bemerkung von Seite 106 aus der Inklusion $\mathfrak{L}_3 \subset \mathfrak{L}_2 \subset \mathfrak{L}_1$. Die in anderen Zeilen verwendeten Einträge (a) bis (d) weisen darauf hin, daß die entsprechenden Aussagen im Anschluß an die Tabelle noch bewiesen werden. Mit [ ] ist eine Aussage gekennzeichnet, deren Beweis in dieser Vorlesung nicht durchgeführt wird.

	Typ				
Problem	3	2	1	0	Begründung
Wort	E	E	E	U	7.1|8.2
Inklusion	E	U	U	U	(a)|8.9
Äquivalenz	E	U	U	U	1.9, 3.3|8.9
Leerheit	E	E	U	U	5.11|8.5
Unendlichkeit	E	E	U	U	5.11|(b)
Ist $L(G) = V_T^*$?	E	U	U	U	1.9, 3.3|8.9
Ist $L(G)$ regulär?	T	U	U	U	trivial|8.9
Ist $L(G)$ kontextfrei?	T	T	U	U	trivial|(c)
Ist G eindeutig?	E	U	-	-	(d)|8.9
Ist $L(G)$ eindeutig?	T	U	-	-	5.1|[ ]
Ist $\complement L(G) = \emptyset$?	E	U	U	U	1.9, 3.3|8.9
Ist $\complement L(G)$ unendlich?	E	U	U	U	1.2, 5.11|8.9
Ist $\complement L(G)$ regulär?	T	U	U	U	1.2|8.9
Ist $\complement L(G)$ kontextfrei?	T	U	U	U	1.2|8.9
Ist $\complement L(G)$ vom selben Typ wie G?	T	U	T	U	1.2|8.9|[ ]|[ ]
Ist $L(G_1) \cap L(G_2) = \emptyset$?	E	U	U	U	1.2, 5.11|8.9
Ist $L(G_1) \cap L(G_2)$ unendlich?	E	U	U	U	1.2, 5.11|8.9
Ist $L(G_1) \cap L(G_2)$ regulär?	T	U	U	U	1.2|8.9
Ist $L(G_1) \cap L(G_2)$ kontextfrei?	T	U	U	U	1.2|8.9
Ist $L(G_1) \cap L(G_2)$ vom selben Typ wie G_1 und G_2?	T	U	T	T	1.2|8.9|9.5|2.6

Wir beweisen die Aussagen (a) bis (d).

Aussage (a): Das Inklusionsproblem für Typ-3-Grammatiken ist entscheidbar.

Beweis: Es seien G_1 und G_2 Typ-3-Grammatiken. Dann existieren nach Satz 3.3 endliche erkennende Automaten E_1 und E_2 mit $L(E_1) = L(G_1)$, $L(E_2) = L(G_2)$. Nach Satz 1.2 können wir einen endlichen erkennenden Automaten D konstruieren mit $L(D) = L(E_1) \cap L(E_2)$. Wegen Satz 1.9 ist entscheidbar, ob $L(E_1) = L(D)$, d.h. $L(E_1) \subset L(E_2)$, gilt. $\square$

Aussage (b): Das Unendlichkeitsproblem für Typ-1-Grammatiken ist unentscheidbar.

Beweis: Wir betrachten die Sprachen $L(PCP)$ und L_{sp} aus Satz 8.8. Beide sind kontextfrei und damit auch vom Typ 1. In Satz 9.5 werden wir zeigen, daß der Durchschnitt zweier Typ-1-Sprachen wieder vom Typ 1 ist. Folglich ist $L' = L(PCP) \cap L_{sp}$ eine Typ-1-Sprache, die durch eine geeignete Typ-1-Grammatik G' erzeugt wird. Nach dem Beweis von Satz 8.9 gilt

$$L(G') = L(PCP) \cap L_{sp} \text{ unendlich} \iff PCP \text{ besitzt eine Lösung.}$$

Aus der Entscheidbarkeit des Unendlichkeitsproblems für Typ-1-Grammatiken würde auch die Entscheidbarkeit des speziellen Postschen Korrespondenzproblems aus Satz 8.7 folgen, ein Widerspruch. $\square$

Aussage (c): Es sei G eine Typ-1-Grammatik. Die Frage, ob $L(G)$ kontextfrei ist, ist unentscheidbar.

Beweis: Nach dem Beweis von (b) ist $L(G') = L(PCP) \cap L_{sp}$ vom Typ 1. Aus dem Beweis der Aussagen (c) und (d) von Satz 8.9 erhalten wir

$$L(G') \text{ kontextfrei} \iff L(PCP) \cap L_{sp} = \emptyset \iff PCP \text{ besitzt keine Lösung.} \quad \square$$

Aussage (d): Es sei G eine Typ-3-Grammatik. Die Frage, ob G eindeutig ist, ist entscheidbar.

Beweis: Zunachst entfernen wir aus G alle unnützen Nichtterminalzeichen, d.h. die Zeichen, die nicht in Ableitungen von X_0 zu Terminalwörtern vorkommen. Diese Konstruktion ist in endlich vielen Schritten durchführbar. Sie hat keinen Einfluß darauf, ob die Grammatik eindeutig ist oder nicht. Für die so gewonnene Grammatik betrachten wir alle Produktionen des Typs $X \to Y$ mit $X, Y \in V_N$ und überprüfen, ob es eine Ableitung $X \Longrightarrow^+ X$ für ein $X \in V_N$ gibt. In diesem Fall ist die Grammatik offensichtlich mehrdeutig. Anderenfalls führt die Konstruktion von Satz 3.2 (dual zu derjenigen aus Satz 3.1) zu einer Typ-3-Grammatik G' mit Produktionen der Form

$$X \to Ya,\ X \to \varepsilon,\ X, Y \in V'_N,\ a \in V_T.$$

Die Konstruktion zeigt, daß G genau dann eindeutig ist, wenn G' eindeutig ist.

Wir müssen zeigen, daß wir für G' die Frage der Eindeutigkeit entscheiden können. Es sei $w = a_p \dots a_1 \in L(G')$ das kürzeste Wort mit zwei verschiedenen Ableitungen.

Jeder Schritt in einer Ableitung bis auf den letzten produziert genau ein Terminalzeichen, diese sind in den beiden Ableitungen gleich. Beide Ableitungen haben folglich dieselbe Länge $p+1$ und müssen sich in den Nichtterminalzeichen unterscheiden. Es gelte also

$$\begin{aligned} D_1: \; X_0 &\Longrightarrow X_1a_1 \Longrightarrow \ldots \Longrightarrow X_{i-1}a_{i-1}\ldots a_1 \Longrightarrow X_ia_i\ldots a_1 \Longrightarrow \ldots \Longrightarrow X_ja_j\ldots a_1 \\ &\Longrightarrow X_{j+1}a_{j+1}\ldots a_1 \Longrightarrow \ldots \Longrightarrow X_pa_p\ldots a_1 \Longrightarrow w, \\ D_2: \; X_0 &\Longrightarrow X_1a_1 \Longrightarrow \ldots \Longrightarrow X_{i-1}a_{i-1}\ldots a_1 \Longrightarrow Y_ia_i\ldots a_1 \Longrightarrow \ldots \Longrightarrow Y_ja_j\ldots a_1 \\ &\Longrightarrow X_{j+1}a_{j+1}\ldots a_1 \Longrightarrow \ldots \Longrightarrow X_pa_p\ldots a_1 \Longrightarrow w \end{aligned}$$

mit $X_i \neq Y_i$, $X_j \neq Y_j$ für feste Indizes $i, j \in \mathbb{N}$, $1 \leq i \leq j \leq p$. Es sei n die Anzahl der Nichtterminalzeichen von G'. Gäbe es innerhalb der Teilableitungen $X_0 \Longrightarrow^* X_{i-1}a_{i-1}\ldots a_1$ oder $X_{j+1}a_{j+1}\ldots a_1 \Longrightarrow^* X_pa_p\ldots a_1$ eine Wiederholung eines Nichtterminalzeichens, so würden wir zwei verschiedene Ableitungen D_1' und D_2' eines Wortes $w' \in L(G')$ mit $|w'| < |w|$ konstruieren können, ein Widerspruch. Folglich gilt

$$i \leq n \text{ und } p - j \leq n.$$

In den Teilableitungen $X_ia_i\ldots a_1 \Longrightarrow^* X_ja_j\ldots a_1$ von D_1 und $Y_ia_i\ldots a_1 \Longrightarrow^* Y_ja_j\ldots a_1$ von D_2 dürfen aus dem gleichen Grund keine zwei Paare $(X_{i'}, Y_{i'})$ und $(X_{i''}, Y_{i''})$, $i' \neq i''$, $i \leq i', i'' \leq j$, übereinstimmen. Es gilt somit

$$j - i + 1 \leq n^2.$$

Es folgt $p+1 \leq n^2+2n$. Die endlich vielen Wörter der Länge $\leq n^2+2n-1$ können nun daraufhin untersucht werden, ob eines von ihnen mehrere Ableitungen besitzt. □

Nach unserer Tabelle ist für eine beliebige Typ-0-Grammatik G unentscheidbar, ob $\complement L(G)$ vom Typ 0 ist. Wäre dies Problem entscheidbar, so würde nach der Churchschen These und Satz 7.4 auch entscheidbar sein, ob $L(G)$ rekursiv ist. Das ist jedoch unentscheidbar, was in dem Buch von *Hopcroft und Ullman* ([15], Example 8.3, Seite 186) gezeigt wird.

8.5 Weitere Unentscheidbarkeitsaussagen für kontextfreie Grammatiken

Wir betrachten jetzt einen etwas anderen Problemkreis. Gibt es für kontextfreie Grammatiken Algorithmen, mit deren Hilfe die Grammatiken so umgeformt werden können, daß sie eine minimale Anzahl von Nichtterminalsymbolen oder eine minimale Anzahl von Produktionen enthalten? Solche Algorithmen wären sehr nützlich, leider existieren sie nicht.

Zum Beweis der Unentscheidbarkeit dieser Probleme benötigen wir die folgenden drei Sätze als Hilfe.

Satz 8.10 Für $V_3 = \{a, b, c\}$ ist es unentscheidbar, ob eine von einer beliebigen kontextfreien Grammatik erzeugte Sprache gleich $V_3^*\{d\}$ ist.

Beweis: Nach dem Beweis von Satz 8.9 ist es unentscheidbar, ob eine beliebige kontextfreie Sprache gleich V_3^* ist. Für eine beliebige kontextfreie Grammatik $G = (V_N, V_T, X_0, F)$ konstruieren wir eine kontextfreie Grammatik

$$G_d = (V_N \cup \{X_0'\}, V_T \cup \{d\}, X_0', F \cup \{X_0' \to X_0 d\}).$$

Dann erhalten wir

$$L(G) = V_3^* \iff L(G_d) = V_3^*\{d\},$$

woraus die Behauptung des Satzes folgt. □

Satz 8.11 Die Grammatik

$$G_1 = (\{X_0\}, \{a, b, c, d\}, X_0, \{X_0 \to aX_0, X_0 \to bX_0, X_0 \to cX_0, X_0 \to d\})$$

ist eine kontextfreie Grammatik mit minimaler Anzahl von Produktionen bezüglich aller kontextfreien Grammatiken für die Sprache $V_3^*\{d\}$. Außerdem ist G_1, abgesehen von Umbenennungen von X_0, die einzige kontextfreie Grammatik mit vier Produktionen, die $V_3^*\{d\}$ erzeugt.

Beweis: Ersichtlich gilt $L(G_1) = V_3^*\{d\}$. Es sei G eine beliebige kontextfreie Grammatik mit $L(G) = V_3^*\{d\}$. Wir betrachten Ableitungen der Wörter

$$ad,\ bd,\ cd,\ d \in V_3^*\{d\}.$$

In diesen Ableitungen müssen Produktionen mit den rechten Seiten

$$P_1 a Q_1,\ P_2 b Q_2,\ P_3 c Q_3 \text{ bzw. } P_4 d P_5$$

mit $P_i \in V_N^*$, $Q_j \in (V_N \cup \{d\})^*$, $i = 1, \ldots, 5$, $j = 1, 2, 3$, auftreten. Somit enthält G mindestens 4 Produktionen. Wir nehmen nun an, daß G genau 4 Produktionen besitzt. Dann besitzt G keine Produktionen mit ε auf der rechten Seite. Nichtterminalzeichen können also nicht gelöscht werden. Es folgt $P_i = \varepsilon$ für $i = 1, \ldots, 5$, und für Q_j gibt es nur die Möglichkeiten

$$Q_j = d \ \vee \ Q_j = \varepsilon \ \vee \ Q_j \in V_N.$$

Wegen $P_i = \varepsilon$ gibt es eine Produktion $X \to d$. Dann ist nur $X = X_0$ möglich, anderenfalls wäre d nicht erzeugbar. Ist $Q_1 = d$, so kann $a^2 d \in V_3^*\{d\}$ nicht abgeleitet werden. Für $Q_1 = \varepsilon$ kann $ad \in V_3^*\{d\}$ und für $Q_1 = X \neq X_0$ kann $ad \in V_3^*\{d\}$ nicht erzeugt werden. Somit bleibt nur die Möglichkeit $Q_1 = X_0$. Analog zeigen wir $Q_2 = Q_3 = X_0$. Offenbar bestehen die linken Seiten der Produktionen aus X_0, weil man sonst von X_0 aus nicht alle Wörter von $V_3^*\{d\}$ erhalten würde. □

Satz 8.12 Es sei $G = (\{X_0\}, V_3 \cup \{d\}, X_0, F)$ eine kontextfreie Grammatik mit $L(G) = V_3^*\{d\}$. Dann sind alle Produktionen von F von einer der beiden folgenden Formen:

$$X_0 \to wX_0,\ X_0 \to wd \text{ mit } w \in V_3^*.$$

Beweis: Wegen $\varepsilon \notin V_3^*\{d\}$ folgt $(X_0, \varepsilon) \notin F$ und damit $(X_0, d) \in F$. Da das Zeichen d in den Wörtern von $V_3^*\{d\}$ ganz hinten vorkommt, darf das Nichtterminalsymbol X_0 auf der rechten Seite einer Produktion höchstens ganz hinten auftreten. Somit sind die Produktionen, die nur Terminalzeichen auf der rechten Seite haben, von der Form $X_0 \to wd$ mit $w \in V_3^*$, andere Produktionen sind von der Form $X_0 \to wX_0$ mit $w \in V_3^*$. □

Satz 8.13 Es sei G eine beliebige kontextfreie Grammatik.

(a) Dann existiert kein Algorithmus, der eine zu G äquivalente kontextfreie Grammatik mit einer Minimalzahl von Nichtterminalzeichen konstruiert (Nichtterminalzeichenminimierungsproblem).

(b) Dann existiert kein Algorithmus, der eine zu G äquivalente kontextfreie Grammatik mit einer Minimalzahl von Produktionen konstruiert (Produktionenminimierungsproblem).

Beweis: Wir nehmen an, das Problem (a) sei entscheidbar. Für eine beliebige kontextfreie Grammatik G sei G_{nt} eine kontextfreie Grammatik mit einer Minimalzahl von Nichtterminalsymbolen und $L(G) = L(G_{nt})$.

Enthält G_{nt} mindestens 2 Nichtterminalzeichen, so folgt nach Satz 8.11 die Gültigkeit von $L(G_{nt}) \neq V_3^*\{d\}$, da G_1 nur ein Nichtterminalzeichen besitzt. Auch in dem Fall, daß G_{nt} genau ein Nichtterminalzeichen hat und Produktionen besitzt, die nicht von der Form aus Satz 8.12 sind, ist $L(G_{nt}) \neq V_3^*\{d\}$. Es bleibt der Fall übrig, daß G_{nt} genau ein Nichtterminalzeichen besitzt und die Produktionen von der Form aus Satz 8.12 sind. Das bedeutet, daß G_{nt} rechtslinear ist. Nach Satz 1.9 ist dann entscheidbar, ob $L(G_{nt}) = V_3^*\{d\}$ gilt oder nicht. Die Betrachtung aller möglichen Fälle zeigt also im Widerspruch zu Satz 8.10, daß die Frage $L(G) = V_3^*\{d\}$ entscheidbar ist.

Wir nehmen nun an, daß das Problem (b) entscheidbar ist. Für eine beliebige kontextfreie Grammatik G sei G_{pr} eine kontextfreie Grammatik mit einer Minimalzahl von Produktionen und $L(G) = L(G_{pr})$. Ist die Anzahl der Produktionen in $G_{pr} \neq 4$, so folgt nach Satz 8.11 die Ungleichung $L(G_{pr}) \neq V_3^*\{d\}$. Ist die Anzahl gleich 4, so ist nach Satz 8.11 entscheidbar, ob $L(G_{pr}) = V_3^*\{d\}$ gilt, da G_1 die einzige derartige Grammatik ist. Insgesamt ist also im Widerspruch zu Satz 8.10 die Frage $L(G) = L(G_{pr}) = V_3^*\{d\}$ entscheidbar. □

8.6 Ein unentscheidbares Problem für Matrizen

Das Postsche Korrespondenzproblem wird häufig auch für den Beweis der Unentscheidbarkeit von Problemen aus ganz anderen Bereichen benutzt. Als ein Beispiel betrachten wir das Sterblichkeitsproblem für Matrizen.

Definition 8.3 Es sei I eine endliche Indexmenge und $\{M_i \mid i \in I\}$ eine Menge von 3×3-Matrizen mit Elementen aus $\mathbb{Z}$. $\{M_i \mid i \in I\}$ heißt *sterblich* bezüglich (j_1, j_2), $1 \leq j_1, j_2 \leq 3$, wenn $k \in \mathbb{N}$ und $i_\kappa \in I$, $1 \leq \kappa \leq k$, existieren, so daß das Element

(j_1, j_2) von $M = M_{i_1} \cdots M_{i_k}$ (das Element der j_1-ten Zeile und j_2-ten Spalte) gleich 0 ist. $\square$

Satz 8.14 Das Sterblichkeitsproblem für Matrizen ist unentscheidbar.

Beweis: Es sei $PCP = (V, n, \alpha, \beta)$ ein beliebiges Postsches Korrespondenzproblem. Der Beweis ist geführt, wenn wir zu PCP eine Menge von 3×3-Matrizen mit der Eigenschaft

$$PCP \text{ besitzt eine Lösung} \iff \{M_i \mid i \in I\} \text{ sterblich bezüglich (3,2)}$$

angeben können.

Wir werden zu $u, v \in V^*$ eine 3×3-Matrix $M(u, v)$ konstruieren, die die folgenden Eigenschaften hat:

$$(1) \quad \text{Das Element (3,2) von } M(u,v) \text{ ist } 0 \iff u = v.$$
$$(2) \quad \bigwedge_{u_1, u_2, v_1, v_2 \in V^*} M(u_1, v_1) \cdot M(u_2, v_2) = M(u_1 u_2, v_1 v_2).$$

Wenn wir dies gezeigt haben, dann setzen wir speziell $M_i = M(\alpha_i, \beta_i)$, $i = 1, \ldots, n$, und erhalten damit die Matrizenmenge $\{M_i \mid i = 1, \ldots, n\}$. Aus (1) und (2) folgt

$$\begin{aligned} PCP \text{ besitzt eine Lösung} &\iff \text{das Element (3,2) von } M_{i_1} \cdots M_{i_k} \text{ ist gleich } 0 \\ &\iff \{M_i \mid i = 1, \ldots, n\} \text{ sterblich bezüglich (3,2).} \end{aligned}$$

Wir müssen also noch die Matrizen $M(u, v)$ mit den Eigenschaften (1) und (2) konstruieren. Zunächst werden Abbildungen $\gamma, \delta : V^* \to \mathbb{N}_0$ für $w \in V^*$ durch

$$\gamma(w) = i \text{ und } \delta(w) = l^{|w|}$$

definiert, wobei $l = \text{card}(V)$ gilt und w das i-te Wort der Aufzählung der Wörter aus V^* ist. Dabei sind die Wörter der Länge nach und bei gleicher Länge lexikographisch geordnet. Speziell sei ε das 0-te Wort. Für $w, z \in V^*$ gilt

$$\begin{aligned} \gamma(wz) &= \gamma(w)\delta(z) + \gamma(z), \\ \delta(wz) &= \delta(w)\delta(z). \end{aligned}$$

Aus $l^{|wz|} = l^{|w|+|z|} = l^{|w|} l^{|z|}$ folgt die zweite Gleichung. Die erste Gleichung wird durch vollständige Induktion über die Länge von z bewiesen. Für $|z| = 0$ ist sie wegen

$$\gamma(w) = \gamma(w) \cdot 1 + 0$$

erfüllt. Für $|z| = 1$ setzen wir $z = x \in V$ und betrachten die Aufzählung der Wörter bis w:

0	1						$\gamma(w)$
ε,	v_1,	$\ldots$,	v_l,	$v_1 v_1$,	$\ldots$,	w',	w.

Durch jeweiliges Anhängen der Symbole aus V erhalten wir eine Aufzählung der Wörter bis $w'v_l$:

$$\begin{array}{cccccccc} 0 & 1 & & & & & \gamma(w)l & \\ \varepsilon, & \underbrace{v_1,\ldots,v_l}_{\varepsilon}, & \underbrace{v_1v_1,\ldots,v_1v_l}_{v_1}, & \ldots, & \underbrace{w'v_1,\ldots,w'v_l}_{w'}, & wv_1,\ldots,wx. \end{array}$$

Es folgt $\gamma(wx) = \gamma(w)\cdot l + \gamma(x)$. Wir nehmen nun an, daß die erste Gleichung für alle z mit $|z| \leq k$, $k \geq 1$, erfüllt ist. Es sei $z' = zx$ mit $x \in V$. Unter Benutzung der Induktionsannahme erhalten wir

$$\begin{array}{rcl} \gamma(wz') = \gamma((wz)x) & = & \gamma(wz)\delta(x) + \gamma(x) \\ & = & (\gamma(w)\delta(z) + \gamma(z))\delta(x) + \gamma(x) \\ & = & \gamma(w)\delta(z)\delta(x) + \gamma(z)\delta(x) + \gamma(x) \\ & = & \gamma(w)\delta(zx) + \gamma(zx) \\ & = & \gamma(w)\delta(z') + \gamma(z'), \end{array}$$

was zu beweisen war.

Für $u, v \in V^*$ wird jetzt

$$M(u,v) = \begin{pmatrix} \delta(u) & \delta(v)-\delta(u) & 0 \\ 0 & \delta(v) & 0 \\ \gamma(u) & \gamma(v)-\gamma(u) & 1 \end{pmatrix}$$

gesetzt. $M(u,v)$ hat die gewünschten Eigenschaften, denn das Element (3,2) ist genau dann gleich 0, wenn $\gamma(v)-\gamma(u) = 0$ ist, was genau für $u = v$ erfüllt ist. Damit ist (1) bewiesen.

Für beliebige $u_1, u_2, v_1, v_2 \in V^*$ wird (2) mit Hilfe der Gleichungen für γ und δ durch

$$\begin{aligned} & M(u_1,v_1)\cdot M(u_2,v_2) \\ & = \begin{pmatrix} \delta(u_1) & \delta(v_1)-\delta(u_1) & 0 \\ 0 & \delta(v_1) & 0 \\ \gamma(u_1) & \gamma(v_1)-\gamma(u_1) & 1 \end{pmatrix} \cdot \begin{pmatrix} \delta(u_2) & \delta(v_2)-\delta(u_2) & 0 \\ 0 & \delta(v_2) & 0 \\ \gamma(u_2) & \gamma(v_2)-\gamma(u_2) & 1 \end{pmatrix} \\ & = \begin{pmatrix} \delta(u_1)\delta(u_2) & \delta(v_1)\delta(v_2)-\delta(u_1)\delta(u_2) & 0 \\ 0 & \delta(v_1)\delta(v_2) & 0 \\ \gamma(u_1)\delta(u_2)+\gamma(u_2) & (\gamma(v_1)\delta(v_2)+\gamma(v_2))-(\gamma(u_1)\delta(u_2)+\gamma(u_2)) & 1 \end{pmatrix} \\ & = M(u_1u_2, v_1v_2) \end{aligned}$$

bewiesen. $\square$

Kapitel 9

Charakterisierung von Typ-1-Sprachen

9.1 Ein Normalformsatz für Typ-1-Grammatiken

Wie für kontextfreie Grammatiken existieren auch für kontextsensitive Grammatiken Normalformen. Wir definieren zunächst Normalformen für monotone Grammatiken. Mit solchen Normalformen lassen sich alle ε-freien kontextsensitiven Sprachen erzeugen.

Definition 9.1 Es sei $G = (V_N, V_T, X_0, F)$ eine Grammatik. G ist in *Kuroda-Normalform*, wenn alle Produktionen von G die Form

$$X \to a,\ X \to YZ,\ XY \to ZW \text{ mit } X, Y, Z, W \in V_N, a \in V_T$$

besitzen. □

Satz 9.1 Es sei $G = (V_N, V_T, X_0, F)$ eine ε-freie kontextsensitive Grammatik. Dann kann eine äquivalente Grammatik in Kuroda-Normalform konstruiert werden.

Beweis: Zunächst wollen wir Produktionen

$$X \to Y \text{ mit } X, Y \in V_N$$

entfernen. Dies geschieht auf dieselbe Art wie im Beweis von Satz 5.4. Wir können also ein $k \in \mathbb{N}$ sowie Mengen $U_k(x)$ für alle $x \in V = V_N \cup V_T$ konstruieren, die alle Zeichen $y \in V$ enthalten, die man in höchstens $k-1$ Schritten aus x ableiten kann und wobei außerdem $U_k(x) = U_\nu(x)$ für alle $\nu \in \mathbb{N}$ gilt. Speziell für $a \in V_T$ gilt

$$a \in L(G) \iff a \in U_k(X_0).$$

Eine äquivalente Grammatik $G_1 = (V_N, V_T, X_0, F')$ ergibt sich durch die folgende Angabe der Produktionenmenge F':

(1) $(X_0, a) \in F'$ für alle $a \in V_T \cap U_k(X_0)$.

(2) $(x_1 \dots x_m, y_1 \dots y_n) \in F'$ mit $x_1 \dots x_m \in V^* V_N V^*$, $x_1, \dots, x_m, y_1, \dots, y_n \in V$, $1 \leq m \leq n$, $n \geq 2$, für die eine Produktion $(\bar{x}_1 \dots \bar{x}_m, \bar{y}_1 \dots \bar{y}_n) \in F$ existiert mit $\bar{x}_\mu \in U_k(x_\mu)$, $y_\nu \in U_k(\bar{y}_\nu)$ für alle $\mu = 1, \dots, m$ and $\nu = 1, \dots, n$.

Nach Satz 2.2 konstruieren wir zu G_1 eine äquivalente Grammatik $G_2 = (V_N', V_T, X_0, F'')$, bei der die einzigen Produktionen in G_2, die Terminalzeichen enthalten, von der Form $X \to a$ mit $X \in V_N'$ und $a \in V_T$ sind. Produktionen, die noch nicht die gewünschten Eigenschaften des Satzes haben, haben die Gestalt

$$X_1 X_2 \dots X_m \to Y_1 Y_2 \dots Y_n \text{ mit } 1 \leq m \leq n,\ n \geq 3.$$

Die Produktionen des Falls $m = 1$ mit $n \geq 3$ werden mit Hilfe jeweils neuer Nichtterminalzeichen $Z_1, \dots, Z_{n-2}$ durch

$$X_1 \to Y_1 Z_1,\ Z_1 \to Y_2 Z_2,\ \dots, Z_{n-2} \to Y_{n-1} Y_n$$

ersetzt. Im Fall $m \geq 2$ geschieht das durch die folgende Menge von Produktionen der gewünschten Form:

$$\begin{array}{lcl}
X_1 X_2 & \to & Y_1 Z_2, \\
Z_2 X_3 & \to & Y_2 Z_3, \\
 & \vdots & \\
Z_{m-1} X_m & \to & Y_{m-1} Z_m, \\
Z_m & \to & Y_m Z_{m+1}, \\
Z_{m+1} & \to & Y_{m+1} Z_{m+2}, \\
 & \vdots & \\
Z_{n-1} & \to & Y_{n-1} Y_n.
\end{array}$$

Man beachte, daß bereits links von dem jeweiligen Zeichen Z_i vor dem Abschluß der Simulation der gegebenen Produktion eventuell andere Ableitungsschritte gestartet werden können, ggf. unter Benutzung von Teilwörtern von $Y_1 \dots Y_{i-1}$. Ohne Änderung der erzeugten Sprache kann dieser Start jedoch hinter das Ende der Simulation verschoben werden. Ähnliche Überlegungen gelten für Ableitungsschritte rechts von dem Z_i. Benutzen diese sogar Teilwörter von $X_{i+1} \dots X_m$, dann müssen diese Schritte, wenn sie überhaupt erfolgreich abgeschlossen werden sollen, den richtigen Kontext wiederherstellen, und sie hätten folglich schon vor der Simulierung vollständig durchgeführt werden können. □

Eine Normalform von kontextsensitiven Grammatiken wird durch die folgende Definition gegeben.

Definition 9.2 Es sei $G = (V_N, V_T, X_0, F)$ eine kontextsensitive Grammatik. G ist in *linksseitiger Normalform*, wenn alle Produktionen von G die Form

$$X \to a,\ X \to YZ,\ XY \to XZ \text{ mit } X, Y, Z \in V_N, a \in V_T$$

besitzen. □

In [18] findet sich ein langer Beweis für die folgende Aussage.

Satz 9.2 Es sei G eine ε-freie kontextsensitive Grammatik. Dann kann eine äquivalente kontextsensitiven Grammatik in linksseitiger Normalform konstruiert werden. □

9.2 Der Platzbedarfsatz für Typ-1-Grammatiken

In einer Ableitung eines Wortes w gemäß einer kontextsensitiven Grammatik kommt kein Wort vor, das länger als w ist, d.h., der Platzbedarf in der Ableitung ist nicht größer als w. Es soll untersucht werden, wie diese Platzbedarfbedingung für Typ-0-Grammatiken erweitert werden kann, so daß die erzeugten Sprachen vom Typ 1 sind.

Definition 9.3 Es sei $G = (V_N, V_T, X_0, F)$ eine Grammatik und

$$D: \; X_0 = P_0 \Longrightarrow P_1 \Longrightarrow \ldots \Longrightarrow P_n = P$$

eine Ableitung von $P \in (V_N \cup V_T)^*$ gemäß G. Dann heißt

$$PB_G(P, D) = \max\{|P_i| \mid 0 \leq i \leq n\}$$

der *Platzbedarf von P bei der Ableitung D*. Weiter ist

$$PB_G(P) = \min\{PB_G(P, D) \mid D \text{ Ableitung von } P\}$$

der *Platzbedarf von P*. □

Ersichtlich gilt $PB_G(P) \geq |P|$ für alle G und P.

Satz 9.3 Es sei $G = (V_N, V_T, X_0, F)$ eine Typ-0-Grammatik, und es existiere ein $p \in \mathbb{N}$, so daß für alle $w \in L(G)$, $w \neq \varepsilon$, die Relation $PB_G(w) \leq p \cdot |w|$ gilt. Dann folgt $L(G) \in \mathfrak{L}_1$.

Beweis: Ohne Beschränkung der Allgemeinheit können wir annehmen, daß $\varepsilon \notin L(G)$ gilt. Anderenfalls ersetzen wir G wie im Beweis von Satz 2.3 (Abschluß von $\mathfrak{L}_0$ unter Iteration) durch eine ε-freie Grammatik $\bar{G}$ mit $L(\bar{G}) = L(G) - \{\varepsilon\}$. Ist $L(\bar{G})$ kontextsensitiv, dann ist es nach Satz 2.3 auch $L(G) = L(\bar{G}) \cup \{\varepsilon\}$.

Wir führen den Beweis durch vollständige Induktion über p. Es sei $p = 1$. Nach der Annahme des Satzes gelte also

$$(1) \qquad PB_G(w) = |w| \text{ für alle } w \in L(G).$$

Wir konstruieren die Grammatik $G' = (V_N \cup \{Y\}, V_T, X_0, F')$, wobei

$$\begin{aligned} F' \;=\; & \{(P, Y^i Q) \mid (P, Q) \in F,\ |P| - |Q| = i > 0\} \\ & \cup \{(Y^i P, Q) \mid (P, Q) \in F,\ |P| \leq |Q|,\ i = 0, 1, \ldots, |Q| - |P|\} \\ & \cup \{(Y\alpha, \alpha Y), (\alpha Y, Y\alpha) \mid \alpha \in V_N \cup V_T\} \end{aligned}$$

gilt. Offenbar ist G' monoton. Nach Satz 2.7 ist dann $L(G')$ kontextsensitiv. Es muß gezeigt werden, daß $L(G) = L(G')$ gilt. Wir sehen sofort, daß höchstens Wörter aus $L(G)$ durch G' erzeugt werden können. Umgekehrt kann jede Ableitung in G durch eine Ableitung in G' wie folgt simuliert werden. Bei der erstmaligen Anwendung einer Produktion aus der ersten Teilmenge von F' wird das Symbol Y eingeführt. Wegen (1) bleibt dabei die Länge des entsprechenden Wortes der Ableitung $\leq |w|$. Bei weiterer Anwendung von Produktionen aus derselben Teilmenge gilt das gleiche. Vor Anwendung einer Produktion aus der zweiten Teilmenge werden die Y durch Produktionen aus der dritten Teilmenge vor das entsprechende P_1 der Produktion $(P_1, Q_1) \in F$ gebracht. Falls in dem betrachteten Teilwort i Symbole Y, $i \leq |Q_1| - |P_1|$, vorkommen, so werden diese durch die Wahl der Produktion $(Y^i P_1, Q_1) \in F'$ alle gelöscht, anderenfalls werden genau $|Q_1| - |P_1|$ Vorkommen von Y gelöscht. Wegen (1) bleibt die Länge des neu erzeugten Wortes der Ableitung $\leq |w|$. Wir stellen fest, daß jedes Wort der so gewählten Ableitung eine Länge $\leq |w|$ hat und jeder Ableitungsschritt in G durch einen oder mehrere Schritte in G' simuliert werden kann. Am Ende sind folglich alle Y gelöscht, wir erhalten also $w \in L(G')$.

Die Induktionsannahme lautet, daß die Aussage des Satzes für $p-1$, $p \geq 2$, erfüllt ist. Es sei G eine Grammatik mit

$$PB_G(w) \leq p \cdot |w|.$$

Wir konstruieren eine äquivalente Grammatik G' mit

$$PB_{G'}(w) \leq (p-1) \cdot |w| \text{ für alle } w \in L(G') = L(G).$$

Die Idee der Konstruktion von G' besteht darin, daß zwei Symbole von G in ein nichtterminales Zeichen von G' gepackt werden. Dadurch kann der Platzbedarf reduziert werden. Um dieses durchführen zu können, wird zunächst eine Hilfsgrammatik G_1 konstruiert, bei der in allen Ableitungen jedes Wort außer dem ersten von gerader Länge ist. Das bedeutet, daß G_1 die Zusammenfassung von zwei Symbolen zu einem erleichtert. Ohne Beschränkung der Allgemeinheit seien die einzigen Produktionen von G, die terminale Zeichen enthalten, von der Form

$$X \to a,\ X \in V_N,\ a \in V_T.$$

Nach Satz 2.2 ist dies immer erreichbar, weil die dortige Konstruktion den Platzbedarf einer Ableitung nicht verändert.

Wir definieren eine Grammatik

$$G_1 = (V_N \cup \{Y_0, C\}, V_T \cup \{c\}, Y_0, F_1),$$

wobei die Produktionenmenge F_1 wie folgt gegeben ist:

$$(a) \quad \bigwedge_{\alpha \in V_N \cup V_T} ((Y_0, CX_0), (\alpha C, C\alpha), (C\alpha, \alpha C) \in F_1), \quad \bigwedge_{a \in V_T} (Ca, ca) \in F_1,$$

$$(b) \quad (P,Q) \in F,\ |P|, |Q| \text{ gerade} \implies (P,Q) \in F_1,$$

$$(c) \quad (P,Q) \in F,\ |P|, |Q| \text{ ungerade} \implies \bigwedge_{X \in V_N \cup \{C\} \cup V_T} (PX, QX), (XP, XQ) \in F_1,$$

$$(d) \quad (P,Q) \in F,\ |P| \text{ gerade},\ |Q| \text{ ungerade} \implies \bigwedge_{X \in V_N \cup V_T} (P, CQ), (CXP, XQ), (CPX, QX) \in F_1,$$

$$(e) \quad (P,Q) \in F,\ |P| \text{ ungerade},\ |Q| \text{ gerade} \implies \bigwedge_{X \in V_N \cup V_T} (CP, Q), (PX, CQX), (XP, CXQ) \in F_1.$$

Wir zeigen einige Eigenschaften von G_1:

(1) Da beide Seiten jeder Produktion in G_1 mit Ausnahme von $Y_0 \to CX_0$ von gerader Länge sind, gilt für alle Ableitungen

$$Y_0 \implies P_1 \implies \ldots \implies P_n \text{ gemäß } G_1,$$

daß $|P_i|$ gerade ist für $i = 1, \ldots, n$.

(2) Wird ein Homomorphismus $h : (V_T \cup \{c\})^* \to V_T^*$ durch

$$h(c) = \varepsilon,\ h(a) = a \text{ für alle } a \in V_T$$

definiert, so folgt $h(L(G_1)) \subset L(G)$.

Dies ergibt sich bei Betrachtung der Ableitung eines terminalen Wortes gemäß G_1 dadurch, daß dieselbe Ableitung mit Ausnahme der Einführung, Eliminierung und Bewegung der Symbole C und c auch gemäß G ausführbar ist und dann das Terminalsymbol c durch h gelöscht wird.

(3) Es gilt

$$\bigwedge_{w \in L(G)} \bigvee_{w^* \in L(G_1)} ((PB_{G_1}(w^*) \leq PB_G(w) + 1) \ \wedge \ (w^* = w \ \vee \ w^* = cw)).$$

Nach (2) folgt dann $h(L(G_1)) = L(G)$.

Zum Beweis von (3) beweisen wir eine stärkere Aussage.

(4) Für eine Ableitung

$$D : \ X_0 \Longrightarrow_G P_1 \Longrightarrow_G \ldots \Longrightarrow_G P_n \text{ mit } P_n \in (V_N \cup V_T)^*$$

existiert eine Ableitung

$$D_1 : \ Y_0 \Longrightarrow_{G_1}^* Q \text{ mit } Q = P_n \text{ oder } Q = CP_n \text{ und } PB_{G_1}(Q, D_1) \leq PB_G(P_n, D)+1.$$

Zunächst zeigen wir, daß aus (4) die Aussage (3) folgt. Es sei $w \in L(G)$. Dann existiert eine Ableitung $D : X_0 \Longrightarrow^* w$. Wir können speziell eine solche mit minimalem Platzbedarf wählen. Dann gibt es nach (4) eine Ableitung $D_1 : Y_0 \Longrightarrow^*_{G_1} Q$ mit $Q = w$ oder $Q = Cw$ und $PB_{G_1}(Q, D_1) \leq PB_G(w)+1$. Dann folgt, und zwar im Fall $Q = Cw$ durch zusätzliche Anwendung einer Produktion $(Ca, ca) \in F_1$, die Existenz eines Wortes

$$w^* \in L(G_1) \text{ mit } w^* = w \text{ oder } w^* = cw \text{ und } PB_{G_1}(w^*) \leq PB_G(w) + 1.$$

Die Aussage (4) wird durch vollständige Induktion über die Länge n der Ableitung in G bewiesen. Für $n = 1$ folgt aus der Ableitung $D : X_0 \Longrightarrow P_1$, daß es die Produktion $(X_0, P_1) \in F$ gibt. Es sind zwei Fälle zu unterscheiden. Ist $|P_1|$ ungerade, so existiert nach (c) die Produktion $(CX_0, CP_1) \in F_1$. Wir erhalten damit eine Ableitung

$$D_1 : Y_0 \Longrightarrow_{G_1} CX_0 \Longrightarrow_{G_1} CP_1 = Q \text{ mit } PB_{G_1}(Q, D_1) = 1 + |P_1| = 1 + PB_G(P_1, D).$$

Ist $|P_1|$ gerade, so haben wir nach (e) die Produktion $(CX_0, P_1) \in F_1$. Dann ergibt sich die Ableitung

$$D_1 : Y_0 \Longrightarrow_{G_1} CX_0 \Longrightarrow_{G_1} P_1 = Q \text{ mit } PB(Q, D_1) \leq PB_G(P_1, D) + 1.$$

In beiden Fällen ist die Aussage aus (4) erfüllt.

Wir nehmen an, daß (4) für ein festes $n - 1 \geq 1$ gilt. Wir betrachten die Ableitung

$$D : \; X_0 \Longrightarrow_G P_1 \Longrightarrow_G \ldots \Longrightarrow_G P_n, \; P_n \in (V_N \cup V_T)^*.$$

Laut Induktionsannahme existiert eine Ableitung

$$D_1' : Y_0 \Longrightarrow^*_{G_1} Q_1 \text{ mit}$$
$$Q_1 = P_{n-1} \text{ oder } Q_1 = CP_{n-1} \text{ und } PB_{G_1}(Q_1, D_1') \leq PB_G(P_{n-1}, D_2) + 1,$$

wobei D_2 die Teilableitung von D ist, die aus den ersten $n-1$ Schritten von D besteht. Es sei $(R_1, R_2) \in F$ die letzte in D angewendete Produktion. Wir betrachten nun die verschiedenen Teilfälle.

Es sei $Q_1 = P_{n-1}$. Ist $|R_1|$ gerade, so können von den durch $(R_1, R_2) \in F$ bestimmten Produktionen aus (b) oder (d) höchstens $(R_1, R_2) \in F_1$ oder $(R_1, CR_2) \in F_1$ auf Q_1 angewendet werden, je nachdem, ob $|R_2|$ gerade oder ungerade ist. Im ersten Fall erhalten wir eine Ableitung

$$D_1 : Y_0 \Longrightarrow^*_{G_1} Q_1 \Longrightarrow_{G_1} P_n = Q \text{ mit } PB_{G_1}(Q, D_1) = \max\{PB_{G_1}(Q_1, D_1'), |P_n|\}.$$

Wegen $|P_n| \leq PB_G(P_n, D)$ und $PB_G(P_{n-1}, D_2) \leq PB_G(P_n, D)$ folgt $PB_{G_1}(Q, D_1) \leq PB_G(P_n, D) + 1$. Im zweiten Fall ergibt sich unter zusätzlicher Berücksichtigung von Produktionen $(\alpha C, C\alpha)$ aus (a) die Ableitung

$$D_1 : Y_0 \Longrightarrow^*_G Q_1 \Longrightarrow^*_{G_1} CP_n = Q \text{ mit } PB_{G_1}(Q, D_1) = \max\{PB_{G_1}(Q_1, D_1'), |CP_n|\}.$$

Wegen $|CP_n| \leq PB_G(P_n, D) + 1$ gilt dann auch hier $PB_{G_1}(Q, D_1) \leq PB_G(P_n, D) + 1$.

Ist $|R_1|$ ungerade, so folgt nach der Aussage (1), daß XR_1 oder R_1X ein Teilwort von Q_1 für ein geeignetes $X \in V_N \cup V_T$ ist. Wir untersuchen nur den Fall XR_1, der andere kann völlig analog bewiesen werden. Je nachdem, ob R_2 ungerade oder gerade ist, kommen zur Simulation von $(R_1, R_2) \in F$ nur $(XR_1, XR_2) \in F_1$ oder $(XR_1, CXR_2) \in F_1$ in Frage. Die weitere Behandlung erfolgt wie für gerades $|R_1|$. Damit ist der Fall $Q_1 = P_{n-1}$ abgeschlossen.

Es sei jetzt $Q_1 = CP_{n-1}$. Nach Aussage (1) muß dann $|P_{n-1}|$ ungerade sein. Sind $|R_1|$ und $|R_2|$ gerade, so existiert nach (b) die Produktion $(R_1, R_2) \in F_1$ und damit die Ableitung

$$D_1 : Y_0 \Longrightarrow_{G_1}^* Q_1 = CP_{n-1} \Longrightarrow_{G_1} CP_n$$
$$\text{mit } PB_{G_1}(Q, D_1) = \max\{PB_{G_1}(Q_1, D_1'), |CP_n|\}.$$

Es gelten dieselben Schlüsse wie zuvor.

Ist $|R_1|$ gerade und $|R_2|$ ungerade, dann ist wegen $|P_{n-1}|$ ungerade XR_1 oder R_1X ein Teilwort von P_{n-1} für ein geeignetes $X \in V_N \cup V_T$. Außerdem existieren gemäß (d) die Produktionen $(CXR_1, XR_2) \in F_1$, $(CR_1X, R_2X) \in F_1$. Diese sind anwendbar, wenn zuvor durch Produktionen aus (a) das Symbol C nach rechts an die passende Stelle gerückt wurde. Insgesamt erhalten wir eine Ableitung

$$D_1 : Y_0 \Longrightarrow_{G_1}^* Q_1 = CP_{n-1} \Longrightarrow_{G_1}^* P_n,$$

für die wir wie zuvor $PB_{G_1}(Q, D_1) \leq PB_G(P_n, D) + 1$ beweisen können.

Für ungerade $|R_1|$ und $|R_2|$ existiert die Produktion $(CR_1, CR_2) \in F_1$. Sie ist nach der Bewegung des Symbols C nach rechts durch Produktionen nach (c) anwendbar und liefert damit eine Ableitung

$$D_1 : Y_0 \Longrightarrow_{G_1}^* Q_1 \Longrightarrow_{G_1}^* CP_n.$$

Für ungerades $|R_1|$ und gerades $|R_2|$ erhalten wir die Produktion $(CR_1, R_2) \in F_1$ und so die Ableitung

$$D_1 : Y_0 \Longrightarrow_{G_1}^* Q_1 \Longrightarrow_{G_1}^* P_n.$$

In beiden Fällen gelten weitere Überlegungen wie oben. Insgesamt ist also (4) und damit auch (3) erfüllt.

Wir kehren jetzt zur Induktion über p und zur Konstruktion der Grammatik G' zurück. Wir setzen $V = V_N \cup V_T \cup \{c, C\}$ und erhalten damit

$$G' = (\{[a, b] \mid a, b \in V\}, V_T, [C, X_0], F'),$$

wobei F' aus den Produktionen

$[a, b] \to ab$, $[c, a] \to a$ für alle $a, b \in V_T$,
$[\alpha_1, \alpha_2] \ldots [\alpha_{u-1}, \alpha_u] \to [\beta_1, \beta_2] \ldots [\beta_{\nu-1}, \beta_\nu]$ und
$[\alpha, \alpha_1][\alpha_2, \alpha_3] \ldots [\alpha_u, \beta] \to [\alpha, \beta_1][\beta_2, \beta_3] \ldots [\beta_\nu, \beta]$
für $\alpha, \beta \in V$ und jede Produktion $(\alpha_1 \ldots \alpha_u, \beta_1 \ldots \beta_\nu) \in F_1$ mit $\alpha_i, \beta_j \in V$

besteht. Die Produktion $(Y_0, CX_0) \in F_1$ kommt zwar wegen $Y_0 \notin V$ nicht in F' vor, sie wird jedoch durch das Anfangssymbol $[C, X_0]$ von G' berücksichtigt. Aufgrund der Wahl dieses Anfangssymbols entsprechen die Ableitungen von G' den Ableitungen von G_1 mit dem Unterschied, daß das bei G_1 auftretende Terminalzeichen c bei G' durch die Produktion $([c, a], a) \in F'$ gelöscht wird. Mit Hilfe der Aussage (3) folgt

$$L(G') = h(L(G_1)) = L(G).$$

Es sei $w \in L(G') = L(G)$. Dann existiert nach (3) ein Wort $w^* \in L(G_1)$ mit $PB_{G_1}(w^*) \leq PB_G(w) + 1$ und $w^* = w$ oder $w^* = cw$. Nach der Definition von G' ist dann

$$PB_{G'}(w) \leq \max\{|w|, \frac{1}{2} \cdot PB_{G_1}(w^*)\}.$$

Die Produktionen $[a, b] \to ab$ und $[c, a] \to c$ brauchen erst am Ende des Ableitungsprozesses angewandt werden, so daß die angegebene Platzschranke in jedem Fall reicht. Gilt speziell $PB_{G'}(w) = |w|$, so folgt wegen $p \geq 2$

$$PB_{G'}(w) = |w| \leq (p-1) \cdot |w|.$$

Gilt dagegen $PB_{G'}(w) \leq \frac{1}{2} \cdot PB_{G_1}(w^*)$, so erhalten wir aufgrund der durch $PB_G(w) \leq p \cdot |w|$ gegebenen Voraussetzung

$$PB_{G'}(w) \leq \frac{1}{2} \cdot PB_{G_1}(w^*) \leq \frac{1}{2} \cdot (PB_G(w) + 1) \leq \frac{1}{2} \cdot (p \cdot |w| + 1).$$

Speziell für $p = 2$ ergibt sich wegen der Ganzzahligkeit von $PB_{G'}(w)$

$$PB_{G'}(w) \leq |w| = (2-1) \cdot |w|.$$

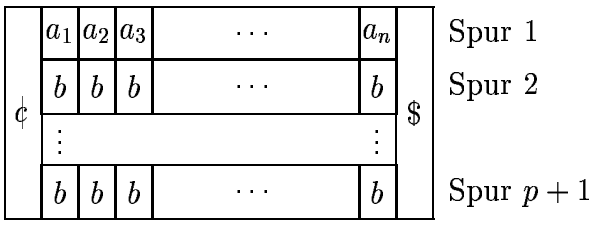
Für $p \geq 3$ folgt

$$PB_{G'}(w) \leq \frac{1}{2} \cdot (p \cdot |w| + 1) = (p-1) \cdot |w| + \frac{1}{2} + (1 - \frac{p}{2}) \cdot |w| \leq (p-1) \cdot |w|. \quad \square$$

Als unmittelbare Folgerung erhalten wir

Satz 9.4 Es sei $L \in \mathfrak{L}_0 - \mathfrak{L}_1$. Für alle $p \in \mathbb{N}$ und alle Grammatiken G mit $L(G) = L$ existiert ein Wort $w \in L$ mit $PB_G(w) > p \cdot |w|$. $\square$

Für Satz 9.3 kann auch ein Beweis angegeben werden, der zu einer Grammatik G mit den Eigenschaften aus dem Satz einen linear beschränkten Automaten B mit $L(B) = L(G)$ konstruiert. Der Beweis verläuft ähnlich den Beweisen von Satz 3.9 und Satz 3.11. Wir geben eine Beweisskizze an. Ohne Beschränkung der Allgemeinheit (siehe Beweis von Satz 9.3) können wir $\varepsilon \notin L(G)$ annehmen. Für ein beliebiges $w \in V_I^+$ können wir durch einen Vor- und Rücklauf des Kopfes eine Unterteilung des Eingabebandes von B in $p+1$ Spuren erreichen:



Dabei hält die erste Spur das Eingabewort $w = a_1 \dots a_n$, während die Felder von Spur 2 bis Spur $p+1$ jeweils mit einem „Blankzeichen" $b \notin V_I$ beschrieben werden. Während der weiteren Arbeit wird die Spur 1 nicht verändert, während auf den anderen p Spuren der Erzeugungsprozeß der Grammatik G simuliert wird. Wegen $PB_G(w) \leq p \cdot |w|$ reicht der Platz auf diesen Spuren aus. Die weiteren Einzelheiten der Simulation erfolgen ähnlich dem Vorgehen in den Beweisen von Satz 3.9 und Satz 3.11. Ein Wort w wird genau dann von B akzeptiert, wenn $w \in L(G)$ gilt.

9.3 Abschlußeigenschaften von Typ-1-Sprachen

Satz 9.5 Die Familie $\mathfrak{L}_1$ ist abgeschlossen unter
(a) Durchschnittsbildung,
(b) ε-freier Substitution und unter ε-freiem Homomorphismus.

Beweis: Zum Beweis von (a) betrachten wir $L, L' \in \mathfrak{L}_1$, die durch Typ-1-Grammatiken G, G' erzeugt werden. Somit gilt $PB_G(w) = |w|$ für alle $w \in L(G)$, eine entsprechende Gleichung ist auch für G' erfüllt. Wir konstruieren eine Grammatik G_1 wie im Beweis von Satz 2.6 auf Seite 30. Die Produktionenmenge von G_1 ist $F \cup F' \cup F_1$. Von den Produktionen aus F_1 erhöht nur die Produktion $Y_0 \to Y_1 X_0 Y_2 X_0' Y_1$ den Platzbedarf, sie kommt jedoch in jeder Ableitung nur genau am Anfang vor. Das bedeutet, daß G_1 die Bedingung aus Satz 9.3 mit $p = 5$ erfüllt. Es folgt $L(G_1) = L \cap L' \in \mathfrak{L}_1$.

Für (b) betrachten wir den Beweis von Satz 2.5 für den Typ 0 und die dort auf Seite 29 angegebenen Grammatiken G, G_i und G_σ. Da wir σ als ε-freie Substitution angenommen haben, gilt $(X_0^i, \varepsilon) \notin F^i$ für $i = 1, \dots, r$. Das Symbol $Y \in V_N^\sigma$ kommt wegen der Gestalt der Produktionen in jedem Wort einer Ableitung gemäß G_σ höchstens einmal vor. Da $Y \to \varepsilon$ die einzige Produktion von G_σ ist, bei der die Länge der linken Seite größer als die der rechten ist, ist der Platzbedarf für eine Ableitung gemäß G_σ durch den Platzbedarf der gegebenen kontextsensitiven Grammatiken, der gleich der Länge des jeweils erzeugten Wortes ist, und einem zusätzlichen Platz für Y gegeben. Wir sehen also, daß die Voraussetzungen von Satz 9.3 mit $p = 2$ erfüllt sind. Es folgt $\sigma(L(G)) = L(G_\sigma) \in \mathfrak{L}_1$. □

Definition 9.4 Es sei $L \subset V^*$ eine Sprache, $k \in \mathbb{N}_0$ und $h : V^* \to V'^*$ ein Homomorphismus. h heißt *k-lineare Löschung bezüglich L*, wenn für alle $w \in L$ die Ungleichung $|w| \leq k \cdot |h(w)|$ erfüllt ist. Eine Familie von Sprachen $\mathfrak{L}$ heißt *abgeschlossen unter linearer Löschung*, wenn $h(L) \in \mathfrak{L}$ gilt für alle $L \in \mathfrak{L}$, $k \in \mathbb{N}_0$ und alle k-linearen Löschungen h bezüglich L. □

Satz 9.6 $\mathfrak{L}_1$ ist abgeschlossen unter linearer Löschung.

Beweis: Es sei $L \in \mathfrak{L}_1$, $k \in \mathbb{N}_0$ und $h : V_T^* \to V_T'^*$ eine k-lineare Löschung bezüglich L. Nach Satz 2.2 wird L von einer Typ-1-Grammatik $G = (V_N, V_T, X_0, F)$ erzeugt, wobei die einzigen Produktionen, die terminale Zeichen enthalten, von der Form $X \to a$ mit $X \in V_N$, $a \in V_T$ sind. Ohne Beschränkung der Allgemeinheit gelte $k \in \mathbb{N}$, denn für $k = 0$ kommen aufgrund der Definition 9.4 nur die Typ-1-Sprachen $\{\varepsilon\}$ und $\emptyset$ in Frage, deren homomorphe Bilder wieder $\{\varepsilon\}$ bzw. $\emptyset$ sind.

Man konstruiere aus F eine Produktionenmenge F', indem jede Produktion der Form $(X, a) \in F$ durch $(X, h(a)) \in F'$ ersetzt wird. Offensichtlich wird $h(L)$ von der Grammatik $G' = (V_N, V_T', X_0, F')$ erzeugt. Es sei $w' \in L(G')$, $w' \neq \varepsilon$, beliebig. Dann existiert $w \in L$ mit $h(w) = w'$. Da h eine k-lineare Löschung bezüglich L ist, folgt $|w| \leq k \cdot |h(w)| = k \cdot |w'|$. Unter Benutzung von $PB_G(w) = |w|$ erhalten wir

$$PB_{G'}(w') = \max\{PB_G(w), |w'|\} = \max\{|w|, |w'|\} \leq k \cdot |w'|.$$

Wegen Satz 9.3 ist daher $L(G')$ vom Typ 1. □

9.4 Zusammenfassung von Abschlußergebnissen

abgeschlossen unter	$\mathfrak{L}_3$	$\mathfrak{L}_2$	$\mathfrak{L}_1$	$\mathfrak{L}_0$	Begründung
Vereinigung	Ja	Ja	Ja	Ja	2.3
Konkatenation	Ja	Ja	Ja	Ja	2.3
Iteration	Ja	Ja	Ja	Ja	2.3
ε-freier Iteration	Ja	Ja	Ja	Ja	2.4
Komplement	Ja	Nein	Ja	Nein	1.2| 5.12|(a)|7.6
Durchschnitt	Ja	Nein	Ja	Ja	1.2|5.12|9.5|2.6
Durchschnitt mit regulären Sprachen	Ja	Ja	Ja	Ja	1.2|5.13|trivial
Spiegelbild	Ja	Ja	Ja	Ja	4.2|trivial
Substitution	Ja	Ja	Nein	Ja	4.2|2.5|(b)|2.5
ε-freier Substitution	Ja	Ja	Ja	Ja	4.2|2.5|9.5|2.5
Homomorphismus	Ja	Ja	Nein	Ja	4.2|2.5|(b)|2.5
ε-freiem Homomorphismus	Ja	Ja	Ja	Ja	4.2|2.5|9.5|2.5
linearer Löschung	Ja	Ja	Ja	Ja	4.2|2.5|9.6|2.5

Der Abschluß von $\mathfrak{L}_1$ unter Komplement (Aussage (a)) wurde erst 1987 gezeigt. Zum Beweis sei auf [27] verwiesen. Für den Beweis der Aussage (b) betrachten wir eine Sprache $L \in \mathfrak{L}_0$. Zu L existiert nach Satz 8.1 eine Typ-1-Sprache L_1, so daß mit dem durch

$$h(a) = h(b) = \varepsilon,\ h(x) = x \text{ für } x \neq a, x \neq b$$

definierten Homomorphismus h die Gleichung $h(L_1) = L$ gilt. Wird speziell $L \in \mathfrak{L}_0 - \mathfrak{L}_1$ gewählt, so folgt $h(L_1) = L \notin \mathfrak{L}_1$. $\mathfrak{L}_1$ ist also nicht unter Homomorphismus und somit auch nicht unter Substitution abgeschlossen.

Kapitel 10

Abstrakte Familien von Sprachen (AFLs)

Wir haben gesehen, daß die in den vorangegangenen Kapiteln eingeführten Sprachfamilien mit wenigen Ausnahmen abgeschlossen waren unter Durchschnitt mit regulären Sprachen, Vereinigungsbildung, Konkatenation und anderen Operationen (siehe etwa Abschnitt 9.4). Dies legt nun nahe, daß man in bezug auf Abschlußeigenschaften nicht spezielle Sprachfamilien wie zum Beispiel die der regulären Sprachen betrachtet, sondern daß man allgemein den Begriff einer abstrakten Familie von Sprachen (abstract family of languages, kurz AFL) einführt, die durch den Abschluß unter gewissen festgelegten Sprachoperationen gekennzeichnet ist. Mit einer solchen AFL kann man weiterarbeiten und z.B. feststellen, daß die gegebenen Abschlußeigenschaften andere Abschlußeigenschaften implizieren. Dann gelten die so gewonnenen Sätze für jede Sprachfamilie, die eine AFL ist. Es ist also nicht nötig, die entsprechenden Beweise für spezielle Sprachfamilien, die AFLs sind, zu wiederholen.

10.1 Gegenseitige Abhängigkeit von Abschlußoperationen

Der Begriff einer Sprachfamilie ist schon oft gebraucht worden (siehe auch Definition 2.6), zum Beispiel die Familie der regulären, der kontextfreien Sprachen usw. Dabei wird nicht angenommen, daß alle Sprachen dasselbe feste Alphabe V_T besitzen. In unendlichen Sprachfamilien kann das „Gesamtalphabet“ aller beteiligten Sprachen beliebig groß werden. Andererseits besitzt jedoch jede Sprache der Familie ein endliches Alphabet. Die folgende Definition legt diese Zusammenhänge genauer fest.

Definition 10.1 Ein Paar $(V, \mathfrak{L})$ heißt eine *Sprachfamilie*, wenn gilt:

(a) V ist eine unendliche Menge (von Symbolen oder Buchstaben),

(b) $\mathfrak{L}$ ist eine Menge von Mengen von Wörtern über V,

(c) für alle $L \in \mathfrak{L}$ existiert eine endliche Teilmenge $V_T \subset V$, so daß $L \subset V_T^*$ gilt (d.h., L ist eine Sprache über V_T),

(d) $\mathfrak{L}$ enthält eine nichtleere Sprache. □

Anstelle von $(V, \mathfrak{L})$ wird im folgenden auch kurz $\mathfrak{L}$ geschrieben. Nach Teil (d) der Definition 10.1 ist weder $\emptyset$ noch $\{\emptyset\}$ eine Sprachfamilie. Jedes Element von $\mathfrak{L}$ ist wegen der Bedingung (c) eine Sprache.

Definition 10.2 Eine Sprachfamilie $(V, \mathfrak{L})$ heißt *abgeschlossen unter Komplementbildung*, wenn mit jeder Sprache L über einem Alphabet V_T auch ihr Komplement $V_T^* - L$ ein Element von $\mathfrak{L}$ ist. □

Bei der Komplementbildung spielt der Bezug zu V_T eine Rolle. Andere Abschlußoperationen hängen häufig nur von der Sprache ab.

Definition 10.3 Eine Sprachfamilie $(V, \mathfrak{L})$ heißt *abgeschlossen* unter einer Operation, die für Sprachen definiert ist, wenn die Operation bei Anwendung auf Sprachen der Familie wieder eine Sprache der Familie liefert. □

Definition 10.4 (a) Eine Sprache L heißt *ε-frei*, wenn $\varepsilon \notin L$ gilt.
(b) Eine Sprachfamilie $(V, \mathfrak{L})$ heißt *ε-frei*, wenn alle $L \in \mathfrak{L}$ ε-frei sind. □

Für eine Sprachfamilie $\mathfrak{L}$ wird auch die Operation der *ε-freien Iteration* betrachtet, also

$$L^+ = \bigcup_{i=1}^{\infty} L^i \quad \text{für } L \in \mathfrak{L}.$$

Definition 10.5 Es sei L eine Sprache über dem Alphabet V_T und $k \in \mathbb{N}$. Ein Homomorphismus $h : V_T^* \to V_T^*$ heißt *k-linear-löschend bezüglich L*, wenn für alle Wörter $w \in L$ die Beziehung $|w| \leq k \cdot |h(w)|$ gilt. □

Dies bedeutet, daß durch k-linear-löschende Homomorphismen nicht zu viel gelöscht wird. Die Länge von $h(w)$ ist durch $\frac{|w|}{k}$ nach unten beschränkt.

Definition 10.6 Es gelte $c \notin V_T$ und $L \subset (V_T\{\varepsilon, c, \ldots, c^{k-1}\})^*$ mit $k \in \mathbb{N}$. Ein Homomorphismus $h : (V_T \cup \{c\})^* \to V_T^*$ heißt *k-beschränkt auf L*, wenn

$$h(c) = \varepsilon \text{ und } h(a) = a \text{ für alle } a \in V_T$$

gilt. □

Da kein Wort $w \in L$ gemäß Definition 10.6 mehr als $k-1$ aufeinanderfolgende Symbole c haben kann, folgt unmittelbar $|w| \leq k \cdot |h(w)|$. Das bedeutet, daß jeder auf L k-beschränkte Homomorphismus auch k-linear-löschend bezüglich L ist.

Definition 10.7 Eine Sprachfamilie $\mathfrak{L}$ heißt *abgeschlossen unter linearer Löschung (beschränktem Homomorphismus)*, wenn $h(L) \in \mathfrak{L}$ für alle $L \in \mathfrak{L}$, $k \in \mathbb{N}$ und k-linear-löschenden (k-beschränkten) Homomorphismen h bezüglich (auf) L gilt. □

Man beachte, daß Definition 10.7, ebenso wie Definition 10.2, nicht durch die allgemeine Definition 10.3 erfaßt wird. Aus Definition 10.7 folgt, daß jede unter linearer Löschung abgeschlossene Sprachfamilie ebenfalls unter beschränktem Homomorphismus abgeschlossen.

Definition 10.8 Es seien V_1 und V_2 Alphabete. Eine Abbildung $g : V_2^* \to \mathfrak{P}(V_1^*)$ heißt *inverser Homomorphismus*, wenn ein Homomorphismus $h : V_1^* \to V_2^*$ existiert, so daß für alle $v \in V_2^*$

$$g(v) = \{w \in V_1^* \mid h(w) = v\}$$

gilt. Es wird $g = h^{-1}$ geschrieben.

Für eine Sprache $L \subset V_2^*$ werde

$$\begin{aligned} g(L) = h^{-1}(L) &= \{w \mid w \in g(v), v \in L\} \\ &= \{w \mid h(w) \in L\} \end{aligned}$$

gesetzt. □

Satz 10.1 Es sei $\mathfrak{L}$ eine Sprachfamilie, die abgeschlossen ist unter Vereinigung, ε-freier Iteration, ε-freiem Homomorphismus, inversem Homomorphismus und unter Durchschnitt mit regulären Sprachen. Dann ist $\mathfrak{L}$ abgeschlossen unter Konkatenation.

Beweis: Für zwei beliebige Sprachen $L_1, L_2 \in \mathfrak{L}$ ist $L_1 L_2 \in \mathfrak{L}$ zu zeigen. Offenbar existiert ein Alphabet V_T mit $L_1, L_2 \subset V_T^*$. Zunächst werden neue Alphabete

$$V_T' = \{a' \mid a \in V_T\}, \quad V_T'' = \{a'' \mid a \in V_T\} \quad \text{und} \quad V = V_T' \cup V_T''$$

definiert. Ein ε-freier Homomorphismus $h : V^* \to V_T^*$ wird dann durch

$$h(a) = h(a') = h(a'') = a \quad \text{für alle } a \in V_T$$

gegeben. Weiter sei L_1' (bzw. L_2') die Sprache, die aus $L_1 - \{\varepsilon\}$ (bzw. $L_2 - \{\varepsilon\}$) entsteht, indem in jedem Wort das erste Symbol a durch a' (bzw. a'') ersetzt wird, d.h.

$$L_1' = h^{-1}(L_1) \cap V_T' V_T^*, \quad L_2' = h^{-1}(L_2) \cap V_T'' V_T^*.$$

Da $\mathfrak{L}$ unter inversem Homomorphismus und unter Durchschnitt mit regulären Sprachen abgeschlossen ist, erhalten wir $L_1', L_2' \in \mathfrak{L}$. Damit folgt unter weiterer Beachtung des Abschlusses von $\mathfrak{L}$ unter Vereinigungsbildung und ε-freier Iteration

$$L_3 = (L_1' \cup L_2')^+ \cap V_T' V_T^* V_T'' V_T^* = \{a' w_1 b'' w_2 \mid a, b \in V_T, a w_1 \in L_1, b w_2 \in L_2\} \in \mathfrak{L}$$

und somit wegen des Abschlusses unter ε-freiem Homomorphismus schließlich $h(L_3) \in \mathfrak{L}$. Wegen

$$L_1 L_2 = \begin{cases} h(L_3), & \text{falls } \varepsilon \notin L_1 \cup L_2 \\ h(L_3) \cup L_1, & \text{falls } \varepsilon \notin L_1, \varepsilon \in L_2 \\ h(L_3) \cup L_2, & \text{falls } \varepsilon \in L_1, \varepsilon \notin L_2 \\ h(L_3) \cup L_1 \cup L_2, & \text{falls } \varepsilon \in L_1 \cap L_2 \end{cases}$$

ist dann auch $L_1 L_2 \in \mathfrak{L}$. □

Satz 10.2 Es sei $\mathfrak{L}$ eine Sprachfamilie, die abgeschlossen ist unter ε-freier regulärer Substitution, beschränktem Homomorphismus und unter Vereinigungs- und Durchschnittsbildung mit regulären Sprachen. Dann ist $\mathfrak{L}$ abgeschlossen unter inversem Homomorphismus.

Für ε-freie Sprachfamilien gilt dieselbe Aussage, ohne daß der Abschluß unter Vereinigung mit regulären Sprachen vorausgesetzt werden muß.

Beweis: Es sei $L \in \mathfrak{L}$ eine Sprache über dem Alphabet V_T und $h : V^* \to V_T^*$ ein Homomorphismus mit

$$V = \{a_1, \ldots, a_r\} \quad \text{und} \quad h(a_i) = w_i \in V_T^* \quad \text{für alle } 1 \leq i \leq r.$$

Wir müssen zeigen, daß $h^{-1}(L) \in \mathfrak{L}$ gilt.

(a) Falls L ε-frei ist, wird zunächst $k = \max\{|w_i| \mid 1 \leq i \leq r\} + 1$ gesetzt und ein neues Alphabet $V' = \{a_1', \ldots, a_r'\}$ gewählt. Wir definieren eine ε-freie reguläre Substitution σ auf V_T durch

$$\sigma(a) = V'^* a V'^*.$$

Eine endliche Sprache wird durch

$$L_1 = \{a_i' w_i \mid 1 \leq i \leq r\}$$

gegeben. Damit setzen wir

$$L_2 = \sigma(L) \cap L_1^*.$$

Wegen des Abschlusses von $\mathfrak{L}$ unter ε-freier regulärer Substitution und unter Durchschnitt mit regulären Sprachen gilt $L_2 \in \mathfrak{L}$. Es werde ein ε-freier Homomorphismus h_1 durch

$$h_1(a_i') = a_i \text{ für } 1 \leq i \leq r \text{ und } h_1(a) = c \text{ für alle } a \in V_T$$

definiert. Da $\mathfrak{L}$ unter ε-freier Substitution abgeschlossen ist, ist $\mathfrak{L}$ erst recht unter ε-freiem Homomorphismus abgeschlossen. Somit folgt $h_1(L_2) \in \mathfrak{L}$. Außerdem gilt $h_1(L_2) \subset (V \cup \{c\})^+$. Wir definieren weiter einen Homomorphismus h_2 durch

$$h_2(c) = \varepsilon \text{ und } h_2(a_i) = a_i \text{ für alle } a_i \in V.$$

h_2 ist nach der Definition von L_2 und der Wahl von k k-beschränkt auf $h_1(L_2)$. Es gilt nun

$$h_2(h_1(L_2)) = h^{-1}(L). \tag{10.1}$$

Dies sehen wir wie folgt ein. Da L ε-frei ist, enthalten beide Seiten von (10.1) nur nichtleere Wörter. Es sei jetzt $v = a_{i_1} \ldots a_{i_n} \in V^+$ mit $n \geq 1$ und $a_{i_j} \in V$. Dann gilt

$$\begin{aligned} v \in h^{-1}(L) &\iff h(v) = w_{i_1} \ldots w_{i_n} \in L \\ &\iff a_{i_1}' w_{i_1} \ldots a_{i_n}' w_{i_n} \in L_2 \\ &\iff v = a_{i_1} \ldots a_{i_n} \in h_2(h_1(L_2)). \end{aligned}$$

Aus (10.1) und dem Abschluß von $\mathfrak{L}$ unter ε-freiem und beschränktem Homomorphismus folgt $h^{-1}(L) \in \mathfrak{L}$.

(b) Falls L nicht ε-frei ist, erhalten wir wegen des Abschlusses unter Durchschnitt mit regulären Mengen $L - \{\varepsilon\} = L \cap V_T^+ \in \mathfrak{L}$. Nach Teil (a) ist somit $h^{-1}(L - \{\varepsilon\}) \in \mathfrak{L}$. Ist h ε-frei, so gilt $h^{-1}(\varepsilon) = \{\varepsilon\}$. Anderenfalls ist $h^{-1}(\varepsilon) = V_1^*$ mit einem Alphabet $V_1 = \{a_i \mid h(a_i) = \varepsilon\} \subset V$. In beiden Fällen ist $h^{-1}(\varepsilon)$ regulär. Aufgrund des Abschlusses unter Vereinigung mit regulären Sprachen folgt

$$h^{-1}(L) = h^{-1}(L - \{\varepsilon\}) \cup h^{-1}(\varepsilon) \in \mathfrak{L}. \quad \square$$

Wir betrachten nun Sprachfamilien, die unter bestimmten Operationen abgeschlossen sind. Die Wahl der Grundoperationen ist natürlich willkürlich, jedoch motiviert durch die Tatsache, daß alle Sprachfamilien $\mathfrak{L}_i$, $i = 0, 1, 2, 3$, der Chomsky-Hierarchie unter diesen Operationen abgeschlossen sind.

Definition 10.9 Eine Sprachfamilie heißt *abstrakte Familie von Sprachen* (englisch: *abstract family of languages*, kurz *AFL*), wenn sie abgeschlossen ist unter jeder der Operationen Vereinigung, ε-freie Iteration, ε-freier Homomorphismus, inverser Homomorphismus und Durchschnitt mit regulären Sprachen.

Eine AFL heißt *voll*, wenn sie unter jedem beliebigen Homomorphismus abgeschlossen ist. $\square$

Wegen Definition 10.1 existiert eine nichtleere Sprache in jeder AFL.

Satz 10.3 (a) Jede AFL ist abgeschlossen unter Konkatenation.
(b) Jede AFL enthält alle ε-freien regulären Sprachen.
(c) Falls eine AFL nicht ε-frei ist, enthält sie alle regulären Sprachen und ist abgeschlossen unter Iteration.

Beweis: (a) Folgt aus Satz 10.1.

(b) Es sei $\mathfrak{L}$ eine beliebige AFL. Dann existiert eine nichtleere Sprache $L \in \mathfrak{L}$, die sogar so gewählt werden kann, daß L ein Wort $P \neq \varepsilon$ enthält. Dies sehen wir wie folgt ein. Wäre $\{\varepsilon\}$ die einzige nichtleere Sprache in $\mathfrak{L}$, so würde mit dem Homomorphismus

$$h : \{a\}^* \to \{a\}^*, \quad h(a) = \varepsilon,$$

nach Definition 10.9

$$h^{-1}(\varepsilon) = \{a\}^* \in \mathfrak{L}$$

folgen, was der Annahme widerspricht.
Es sei R eine beliebige ε-freie reguläre Sprache über $\{a_1, \ldots, a_r\}$. Zu zeigen ist $R \in \mathfrak{L}$. Wegen $R = \{a_1, \ldots, a_r\}^+ \cap R$ und des Abschlusses von $\mathfrak{L}$ unter Durchschnittsbildung mit regulären Sprachen reicht es,

$$\{a_1, \ldots, a_r\}^+ \in \mathfrak{L}$$

zu zeigen. Hierfür werden zunächst mit einem neuen Symbol $c \notin \{a_1, \ldots, a_r\}$ zwei ε-freie Homomorphismen h_1 und h_2 durch

$$h_1(c) = P \text{ und } h_2(a_i) = c \text{ für } 1 \leq i \leq r$$

definiert. Da die Sprache $\{P\}$ regulär ist, folgt wegen des Abschlusses von $\mathfrak{L}$ unter Durchschnitt mit regulären Sprachen $\{P\} = L \cap \{P\} \in \mathfrak{L}$. Da $P \neq \varepsilon$ gilt, ist $\{c\} = h_1^{-1}(\{P\})$, und wegen des Abschlusses von $\mathfrak{L}$ unter inversem Homomorphismus erhalten wir $\{c\} \in \mathfrak{L}$. Der Abschluß unter ε-freier Iteration liefert $\{c\}^+ \in \mathfrak{L}$ und der unter inversem Homomorphismus schließlich $\{a_1, \ldots, a_r\}^+ = h_2^{-1}(\{c\}^+) \in \mathfrak{L}$.

(c) Es sei $L_1 \in \mathfrak{L}$ mit $\varepsilon \in L_1$. Da $\{\varepsilon\}$ regulär ist, folgt $\{\varepsilon\} = L_1 \cap \{\varepsilon\} \in \mathfrak{L}$. Weiter sei R_1 eine beliebige reguläre Sprache. Nach Teil (b) des Beweises gilt $R_1 - \{\varepsilon\} \in \mathfrak{L}$. Falls R_1 ε-frei ist, folgt $R_1 = R_1 - \{\varepsilon\} \in \mathfrak{L}$. Anderenfalls erhalten wir wegen des Abschlusses von $\mathfrak{L}$ unter Vereinigungsbildung $R_1 = (R_1 - \{\varepsilon\}) \cup \{\varepsilon\} \in \mathfrak{L}$.
$\mathfrak{L}$ ist abgeschlossen unter Iteration, da für eine beliebige Sprache $L \in \mathfrak{L}$ wegen

$$L^* = \begin{cases} L^+ \cup \{\varepsilon\}, & \text{falls } L \text{ } \varepsilon\text{-frei} \\ L^+ & \text{sonst} \end{cases}$$

die Beziehung $L^* \in \mathfrak{L}$ folgt. □

Abstrakte Familien von Sprachen können auch auf andere Arten als in Definition 10.9 charakterisiert werden. Bei speziellen Sprachfamilien ist es in der Regel schwierig, den Abschluß unter inversem Homomorphismus nachzuweisen. Diese Abschlußeigenschaft kann jedoch durch andere ersetzt werden. Dies zeigt

Satz 10.4 Jede Sprachfamilie $\mathfrak{L}$, die eine Sprache mit einem nichtleeren Wort enthält und abgeschlossen ist unter Vereinigung, ε-freier Iteration, ε-freier regulärer Substitution, Durchschnitt mit regulären Sprachen und unter beschränktem (beliebigem) Homomorphismus, ist eine (volle) AFL.

Beweis: Da jeder ε-freie Homomorphismus auch eine ε-freie reguläre Substitution ist, bleibt nach Definition 10.9 lediglich der Abschluß unter inversem Homomorphismus zu zeigen.

Es sei $\mathfrak{L}$ zunächst abgeschlossen unter beschränktem Homomorphismus. Falls $\mathfrak{L}$ ε-frei ist, folgt aus der zweiten Teilaussage von Satz 10.2 der Abschluß von $\mathfrak{L}$ unter inversem Homomorphismus. Anderenfalls gelte $\varepsilon \in L \in \mathfrak{L}$, und es sei h ein beliebiger Homomorphismus. Dann folgt nach dem Beweisteil (a) von Satz 10.2, da die dort benötigten Voraussetzungen auch hier erfüllt sind,

$$h^{-1}(L - \{\varepsilon\}) \in \mathfrak{L}.$$

Wegen $\varepsilon \in L$ gilt $h^{-1}(L) = h^{-1}(L - \{\varepsilon\}) \cup h^{-1}(\varepsilon)$. Aufgrund des Abschlusses von $\mathfrak{L}$ unter Homomorphismus bleibt lediglich zu zeigen, daß die nach dem Beweis von Satz 10.2 reguläre Sprache $h^{-1}(\varepsilon)$ ein Element von $\mathfrak{L}$ ist. Da $\mathfrak{L}$ nicht ε-frei ist, existiert eine Sprache $L_1 \in \mathfrak{L}$ mit $\varepsilon \in L_1$, und wegen des Abschlusses unter Durchschnitt mit regulären Sprachen gilt somit $\{\varepsilon\} = L_1 \cap \{\varepsilon\} \in \mathfrak{L}$. Nach Voraussetzung existiert eine Sprache $L_2 \in \mathfrak{L}$ mit $P \in L_2$, $P \neq \varepsilon$, so daß auch $\{P\} = L_2 \cap \{P\} \in \mathfrak{L}$ gilt. Sei a ein beliebiges Symbol und h_1 ein Homomorphismus mit

$$h_1(a) = P.$$

Da $\{P\}$ ε-frei ist, folgt nach Beweisteil (a) von Satz 10.2

$$h_1^{-1}(\{P\}) = \{a\} \in \mathfrak{L}.$$

Es sei weiter R eine beliebige ε-freie reguläre Sprache. Da durch

$$\sigma_R(a) = R$$

eine ε-freie reguläre Substitution gegeben ist, folgt $R \in \mathfrak{L}$. Wegen $\{\varepsilon\} \in \mathfrak{L}$ und dem Abschluß von $\mathfrak{L}$ unter Homomorphismus enthält $\mathfrak{L}$ alle regulären Sprachen, insbesondere die reguläre Sprache $h^{-1}(\varepsilon)$.

Abschließend betrachten wir den Fall, daß $\mathfrak{L}$ abgeschlossen unter beliebigem Homomorphismus ist. Dann ist $\mathfrak{L}$ insbesondere abgeschlossen unter beschränktem Homomorphismus. Nach den vorhergehenden Überlegungen ist $\mathfrak{L}$ daher unter inversem Homomorphismus abgeschlossen. Aus Definition 10.9 folgt sofort, daß $\mathfrak{L}$ eine volle AFL ist. □

Die Sprachfamilie $\mathfrak{L} = \{\{\varepsilon\}, \emptyset\}$ enthält keine Sprache mit einem nichtleeren Wort. Sie erfüllt alle übrigen Voraussetzungen von Satz 10.4. Trotzdem ist sie keine AFL, denn der Homomorphismus $h : \{a\}^* \to \{\varepsilon\}$ mit $h(a) = \varepsilon$ liefert $h^{-1}(\{\varepsilon\}) = \{a\}^* \notin \mathfrak{L}$. $\mathfrak{L}$ ist also nicht unter inversem Homomorphismus abgeschlossen.

Satz 10.5 (a) Jede der Familien $\mathfrak{L}_i$ für $0 \leq i \leq 3$ ist eine AFL.
(b) $\mathfrak{L}_0, \mathfrak{L}_2, \mathfrak{L}_3$ sind volle AFLs, $\mathfrak{L}_1$ ist nicht voll.

Beweis: Die Familien $\mathfrak{L}_0$, $\mathfrak{L}_2$ und $\mathfrak{L}_3$ sind volle AFLs, da sie die Voraussetzungen von Satz 10.4 erfüllen (siehe Zusammenfassung der Abschlußeigenschaften in Abschnitt 9.4). Die Familie $\mathfrak{L}_1$ ist nach Satz 9.6 abgeschlossen unter linearer Löschung und somit nach den Bemerkungen im Anschluß an Definition 10.7 ebenfalls abgeschlossen unter beschränktem Homomorphismus. Nach Satz 10.4 ist $\mathfrak{L}_1$ daher eine AFL. Sie ist jedoch keine volle AFL, da sie nicht abgeschlossen unter beliebigem Homomorphismus ist. Dies sehen wir wie folgt ein (siehe auch Satz 8.1).

Es sei $G = (V_N, V_T, X_0, F)$ eine Grammatik, die *keine* kontextsensitive Sprache erzeugt. Mit neuen Symbolen $A, X_0', a, b \notin V_N \cup V_T$ definieren wir eine Grammatik

$$G' = (V_N \cup \{A, X_0'\}, V_T \cup \{a, b\}, X_0', F'),$$

mit folgenden Produktionen:

$$\begin{array}{ll}
u \to v, & \text{falls } (u, v) \in F \text{ und } |u| \leq |v|, \\
u \to vA^{|u|-|v|}, & \text{falls } (u, v) \in F \text{ und } |u| > |v|, \\
XA \to AX, & \text{falls } X \in V_N \cup V_T \cup \{b\}, \\
X_0' \to bX_0,\ Ab \to ab. &
\end{array}$$

G' ist monoton. Nach Satz 2.7 ist daher $L(G')$ kontextsensitiv. Weiter wird ein Homomorphismus $h : (V_T \cup \{a, b\})^* \to V_T^*$ durch

$$h(x) = x \quad \text{für alle } x \in V_T \quad \text{und} \quad h(a) = h(b) = \varepsilon$$

angegeben. Dann folgt $h(L(G')) = L(G) \notin \mathfrak{L}_1$. $\mathfrak{L}_1$ ist also nicht abgeschlossen unter Homomorphismus. □

Die Familie der k-linearen Sprachen und die der stochastischen Sprachen sind keine AFLs, da diese z.B. nicht abgeschlossen unter Vereinigung sind. Es gibt auch Sprachfamilien, die unter keiner der AFL-Operationen abgeschlossen sind. Diese heißen *Anti-AFLs.*

Falls eine Sprachfamilie unter Substitution abgeschlossen ist, bleibt zum Nachweis der AFL-Eigenschaft nicht mehr viel zu überprüfen. Dies zeigt

Satz 10.6 Jede Sprachfamilie $\mathfrak{L}$, die alle regulären Sprachen enthält und abgeschlossen ist unter Durchschnitt mit regulären Sprachen und Substitution, ist eine volle AFL.

Beweis: Aus dem Abschluß von $\mathfrak{L}$ unter Substitution folgt, daß $\mathfrak{L}$ auch unter beliebigem Homomorphismus abgeschlossen ist. Wegen $\mathfrak{L}_3 \subset \mathfrak{L}$ folgt $\{a\}^+, \{a,b\} \in \mathfrak{L}$ für beliebige Symbole a und b. Es seien $L_1, L_2 \in \mathfrak{L}$ zwei beliebige Sprachen. Eine Substitution σ wird durch

$$\sigma(a) = L_1 \text{ mit } \sigma(b) = L_2$$

definiert. Damit erhalten wir

$$L_1^+ = \sigma(\{a\}^+) \text{ und } L_1 \cup L_2 = \sigma(\{a,b\}).$$

Wegen des Abschlusses von $\mathfrak{L}$ unter Substitution ist $\mathfrak{L}$ abgeschlossen unter Vereinigung und ε-freier Iteration. Speziell ist $\mathfrak{L}$ auch abgeschlossen unter regulärer Substitution. $\mathfrak{L}$ erfüllt also alle Voraussetzungen von Satz 10.4 und ist folglich eine volle AFL. □

Satz 10.7 Eine ε-freie Sprachfamilie $\mathfrak{L}$, die alle ε-freien regulären Sprachen enthält und abgeschlossen ist unter Durchschnitt mit regulären Sprachen, ε-freier Substitution und beschränktem Homomorphismus, ist eine AFL.

Beweis: Aus den Voraussetzungen folgt der Abschluß von $\mathfrak{L}$ unter ε-freier regulärer Substitution, so daß nach Satz 10.4 lediglich der Abschluß unter Vereinigung und ε-freier Iteration zu zeigen ist, was wie im Beweis zu Satz 10.6 geschehen kann. □

Man kann zeigen, daß die fünf definierenden Abschlußeigenschaften einer AFL voneinander unabhängig sind. Keine dieser Eigenschaften folgt aus den jeweils verbleibenden vier anderen.

10.2 Weitere Eigenschaften von AFLs und verwandte Systeme

Wir erinnern uns an die in Definition 3.2 eingeführten endlichen Übersetzungsautomaten. In Definition 3.2 wurden dann EÜA-Abbildungen und inverse EÜA-Abbildungen

definiert. Für ein Wort w, das auch Symbole enthält, die nicht zum Eingabealphabet von U gehören, ist U nicht definiert. Wir können uns Definition 3.2 so erweitert denken, daß in diesen Fällen $U(w) = \emptyset$ gesetzt wird. Entsprechend können dann U und U^{-1} auch auf beliebige Sprachen angewendet werden. Zusätzlich geben wir noch an:

Definition 10.10 (a) Ein endlicher Übersetzungsautomat (V, F) heißt *ε-frei*, wenn für jede Produktion $s_i a \to v s_j$ von F die Beziehung $v \neq \varepsilon$ gilt.
(b) Eine EÜA-Abbildung heißt *ε-frei*, wenn der zugehörige endliche Übersetzungsautomat ε-frei ist. □

Satz 10.8 Es sei T eine binäre Relation über dem Alphabet V. Dann ist die Sprache

$$L = \{b_1 \dots b_n \mid n \geq 2, (b_i, b_{i+1}) \in T, i = 1, \dots, n-1\} \cup V$$

regulär.

Beweis: Wir definieren die reguläre Menge

$$E = \{\varepsilon\} \cup \bigcup_{(b_i, b_j) \in V \times V - T} V^* b_i b_j V^*.$$

Offenbar gilt $L = V^* - E$. Folglich ist L nach Satz 1.2 ebenfalls regulär. □

Satz 10.9 (a) Jede AFL ist abgeschlossen unter ε-freier EÜA-Abbildung.
(b) Jede volle AFL ist abgeschlossen unter beliebiger EÜA-Abbildung.

Beweis: (a) Es sei U ein beliebiger ε-freier endlicher Übersetzungsautomat mit Eingabemenge V_I, Ausgabemenge V_O, Zustandsmenge S, Anfangszustand s_0 und Endzustandsmenge S_1 gemäß Definition 3.1. Die Produktionen der Produktionenmenge F seien von der Form $s_i a \to v s_j$ für gewisse $a \in V_I$, $s_i, s_j \in S$ und $v \in V_O^+$. Weiter sei $\mathfrak{L}$ eine AFL und $L \in \mathfrak{L}$ eine beliebige Sprache. Dann müssen wir $U(L) \in \mathfrak{L}$ zeigen.
Ohne Beschränkung der Allgemeinheit gelte $L \subset V_I^*$, da anderenfalls wegen $U(L) = U(L \cap V_I^*)$ die Sprache $L \cap V_I^*$ betrachtet werden kann, die wegen des Abschlusses einer AFL unter Durchschnitt mit regulären Sprachen ebenfalls ein Element aus $\mathfrak{L}$ ist. Zunächst wird über dem Alphabet

$$V_1 = \{[s_i, a, v, s_j] \mid (s_i a, v s_j) \in F\}$$

eine binäre Relation T durch

$$([s_i, a, v, s_j], [s_i', a', v', s_j']) \in T \quad \Longleftrightarrow \quad s_j = s_i'$$

definiert. Nach Satz 10.8 ist die Menge

$$R = \{\alpha_1 \dots \alpha_n \mid n \geq 2, (\alpha_i, \alpha_{i+1}) \in T, i = 1, \dots, n-1\} \cup V_1$$

regulär. Man beachte, daß die α_i aus V_1 stammen. Wir setzen

$$\begin{aligned} R_2 &= \{\alpha \mid \alpha = [s_0, a, v, s] \in V_1\}, \\ R_3 &= \{\alpha \mid \alpha = [s, a, v, s_j] \in V_1, s_j \in S_1\} \quad \text{und} \\ R_4 &= R_2 \cap R_3. \end{aligned}$$

Diese Mengen sind endlich und somit regulär. Damit definieren wir die regulären Mengen $R_1 = (R_2 V_1^* R_3 \cup R_4) \cap R$ und schließlich

$$R_5 = \begin{cases} R_1 \cup \{\varepsilon\}, & \text{falls } s_0 \in S_1 \\ R_1 & \text{sonst.} \end{cases}$$

Wir erkennen, daß die Wörter von R_5 die Ableitungen gemäß U simulieren. Es sei $h_1 : V_1^* \to V_I^*$ ein ε-freier und $h_2 : V_1^* \to V_O^*$ ein beliebiger Homomorphismus mit

$$h_1([s_i, a, v, s_j]) = a \quad \text{und} \quad h_2([s_i, a, v, s_j]) = v.$$

Wir erhalten $U(L) = h_2(h_1^{-1}(L) \cap R_5)$. Da $\mathfrak{L}$ eine AFL ist, folgt aufgrund des Abschlusses unter ε-freiem Homomorphismus, inversem Homomorphismus und Durchschnitt mit regulären Sprachen $U(L) \in \mathfrak{L}$.

(b) Der Beweis verläuft analog zu Teil (a) mit dem einzigen Unterschied, daß h_2 nicht ε-frei ist, wenn U nicht ε-frei ist. Die Beweisführung wird dadurch jedoch nicht gestört, da jede volle AFL unter beliebigem Homomorphismus abgeschlossen ist. □

Bei dem vorangegangenen Beweis wurden der Abschluß unter Vereinigung und unter ε-freier Iteration nicht benötigt, so daß Satz 10.9 auch für Sprachfamilien gilt, die nur unter den anderen drei Operationen abgeschlossen sind.

Satz 10.10 Jede AFL ist abgeschlossen unter inverser EÜA-Abbildung.

Beweis: Es sei U ein endlicher Übersetzungsautomat wie im Beweis von Satz 10.9. Es sei $\mathfrak{L}$ eine AFL, und $L \in \mathfrak{L}$ sei eine Sprache, für die ohne Beschränkung der Allgemeinheit $L \subset V_O^*$ angenommen werde. Es gilt

$$U^{-1}(L) = \{w \mid w \in U^{-1}(v), v \in L\} = \{w \mid v \in U(w), v \in L\}.$$

h_1, h_2 und R_5 können wie im Beweis von Satz 10.9 konstruiert werden. Dann gilt $U^{-1}(L) = h_1(h_2^{-1}(L) \cap R_5)$. Da h_1 ε-frei ist, folgt $U^{-1}(L) \in \mathfrak{L}$. □

Wir erkennen, daß auch in diesem Beweis der Abschluß unter Vereinigung und ε-freier Iteration nicht benötigt wurde. Eine Sprachfamilie, die nur unter ε-freiem Homomorphismus, inversem Homomorphismus und Durchschnitt mit regulären Mengen abgeschlossen ist, wird *Trio* genannt. Ist sie auch unter beliebigem Homomorphismus abgeschlossen, so heißt sie *volles Trio*.

Aus dem Abschluß unter EÜA-Abbildung folgen andere Abschlußeigenschaften. Dies wird in den folgenden Sätzen gezeigt.

Satz 10.11 Jede AFL ist abgeschlossen unter beschränktem Homomorphismus.

Beweis: Es sei $\mathfrak{L}$ eine AFL, $k \geq 1$ und $L \in \mathfrak{L}$ eine Sprache über $V_T\{\varepsilon, c, \ldots, c^{k-1}\}$. Weiter sei h ein k-beschränkter Homomorphismus mit

$$h(c) = \varepsilon \quad \text{und} \quad h(a) = a \text{ für alle } a \in V_T.$$

Zu zeigen ist $h(L) \in \mathfrak{L}$.

Wir betrachten den ε-freien endlichen Übersetzungsautomaten U, der das Eingabealphabet

$$V_I = \{[a, c^i] \mid a \in V_T, 0 \leq i \leq k-1\},$$

das Ausgabealphabet V_T, das Zustandsalphabet $\{s_0\}$ und Produktionen $s_0[a, c^i] \to as_0$ für alle $[a, c^i] \in V_I$ besitzt. Wir geben einen Homomorphismus h_1 durch

$$h_1([a, c^i]) = ac^i$$

an. Nach Satz 10.9(a) folgt

$$h(L) = U(h_1^{-1}(L)) \in \mathfrak{L}. \quad \square$$

Satz 10.12 (a) Jede AFL ist abgeschlossen unter ε-freier regulärer Substitution.
(b) Jede volle AFL ist abgeschlossen unter regulärer Substitution.

Beweis: (a) Es sei $\mathfrak{L}$ eine AFL und $L \in \mathfrak{L}$ eine Sprache über dem Alphabet $V = \{a_1, \ldots, a_r\}$. Weiter sei σ die durch $\sigma(a_i) = R_i$, $i = 1, \ldots, r$, definierte ε-freie Substitution, wobei jedes R_i eine ε-freie reguläre Sprache über dem Alphabet V_i ist. Wir zeigen $\sigma(L) \in \mathfrak{L}$.

Wir setzen

$$V' = \{a_i' \mid a_i \in V\} \quad \text{und} \quad R = \left(\bigcup_{i=1}^{r} R_i a_i'\right)^*.$$

Offensichtlich ist R regulär. Es werden Homomorphismen h_1 und h_2 auf $(V' \cup V_1 \cup \ldots \cup V_r)^*$ durch

$$h_1(a_i') = a_i,\ h_1(a) = \varepsilon \quad \text{und} \quad h_2(a_i') = c,\ h_2(a) = a$$

für alle $a \in V_1 \cup \ldots \cup V_r$, $1 \leq i \leq r$, definiert, außerdem ein Homomorphismus h_3 auf $(\{c\} \cup V_1 \cup \ldots \cup V_r)^*$ durch

$$h_3(c) = \varepsilon \quad \text{und} \quad h_3(a) = a$$

für alle $a \in V_1 \cup \ldots \cup V_r$. Damit erhalten wir

$$\sigma(L) = h_3(h_2(h_1^{-1}(L) \cap R)), \tag{10.2}$$

was wir wie folgt einsehen.

Für alle nichtleeren Wörter $a_{i_1} \dots a_{i_k} \in L$ mit $1 \leq i_j \leq r$, $j = 1, \dots, k$, gilt

$$h_1^{-1}(a_{i_1} \dots a_{i_k}) = \{Q_{i_1} a'_{i_1} \dots Q_{i_k} a'_{i_k} Q_{i_{k+1}} \mid Q_{i_j} \in (V_1 \cup \dots \cup V_r)^*, j = 1, \dots, k+1\}.$$

Dann folgt

$$h_1^{-1}(a_{i_1} \dots a_{i_k}) \cap R = \{Q_{i_1} a'_{i_1} \dots Q_{i_k} a'_{i_k} \mid Q_{i_j} \in R_{i_j}, j = 1, \dots, k\},$$

woraus sich

$$(*) \qquad h_2(h_1^{-1}(a_{i_1} \dots a_{i_k}) \cap R) = \{Q_{i_1} c \dots Q_{i_k} c \mid Q_{i_j} \in R_{i_j}, j = 1, \dots, k\}$$

und damit schließlich

$$\begin{aligned} h_3(h_2(h_1^{-1}(a_{i_1} \dots a_{i_k}) \cap R)) &= \{Q_{i_1} \dots Q_{i_k} \mid Q_{i_j} \in R_{i_j}, j = 1, \dots, k\} \\ &= R_{i_1} \dots R_{i_k} \\ &= \sigma(a_{i_1} \dots a_{i_k}). \end{aligned}$$

ergibt. Für $\varepsilon \in L$ gilt

$$h_1^{-1}(\varepsilon) = (V_1 \cup \dots \cup V_r)^*,$$

damit also

$$h_1^{-1}(\varepsilon) \cap R = \{\varepsilon\}$$

und somit

$$\begin{aligned} h_3(h_2(h_1^{-1}(\varepsilon) \cap R)) &= \{\varepsilon\} \\ &= \sigma(\varepsilon). \end{aligned}$$

Folglich ist die Gleichung 10.2 bewiesen. h_2 ist ε-frei, und h_3 ist wegen $(*)$ 2-beschränkt auf $h_2(h_1^{-1}(L) \cap R)$, da die R_i ε-frei sind. Nach Satz 10.11 folgt $\sigma(L) \in \mathfrak{L}$.

(b) Es sei $\mathfrak{L}$ eine volle AFL, und die regulären Sprachen aus (a) seien nicht notwendig ε-frei. Dann bleibt (10.2) richtig, wobei jedoch h_3 nicht unbedingt ε-frei sein muß. Da $\mathfrak{L}$ jedoch unter beliebigem Homomorphismus abgeschlossen ist, folgt auch hier $\sigma(L) \in \mathfrak{L}$. □

Die Ergebnisse aus Satz 10.4, Satz 10.11 und Satz 10.12 lassen sich zusammenfassen zu dem folgenden Satz, der eine äquivalente Charakterisierung für AFLs und volle AFLs darstellt.

Satz 10.13 Es sei $\mathfrak{L}$ eine Sprachfamilie, die eine Sprache mit einem nichtleeren Wort enthält, dann gilt:

$\mathfrak{L}$ ist genau dann eine (volle) AFL, wenn $\mathfrak{L}$ abgeschlossen ist unter Vereinigung, ε-freier Iteration (Iteration), ε-freier regulärer Substitution (regulärer Substitution), Durchschnitt mit regulären Sprachen und unter beschränktem Homomorphismus (beliebigem Homomorphismus). □

Wir wissen, daß zu jeder Familie $\mathfrak{L}_i$ für $i = 0, 1, 2, 3$ eine Klasse von erkennenden Akzeptoren gehört, nämlich Turing-Maschinen, linear-beschränkte Automaten, Kellerautomaten bzw. endliche erkennende Automaten. Entsprechend der Einführung von vollen AFLs können nun *abstrakte Familien von Akzeptoren (AFA)* eingeführt werden, und zwar derart, daß zu jeder AFA F die Sprachfamilie, die durch die Akzeptoren in F akzeptiert wird, eine volle AFL ist und es umgekehrt zu jeder AFL $\mathfrak{L}$ eine AFA F gibt, so daß $\mathfrak{L}$ gerade die Sprachfamilie ist, die von den Akzeptoren von F akzeptiert wird.

Eine ausführliche Darstellung von AFLs und AFAs findet sich in dem Buch von *S. Ginsburg* [10].

Literaturverzeichnis

[1] *H. Becker, H. Walter:* Formale Sprachen. Vieweg, Braunschweig 1977.

[2] *W. Brauer:* Automatentheorie. Teubner, Stuttgart 1984.

[3] *W. Bucher, H. Maurer:* Theoretische Grundlagen der Programmiersprachen. Automaten und Sprachen. Bibliographisches Institut. Mannheim 1984.

[4] *V. Claus:* Stochastische Automaten. Teubner, Stuttgart 1971.

[5] *J. Dassow, Gh. Păun:* Regulated Rewriting in Formal Language Theory. Akademie-Verlag, Berlin 1989 und Springer, Berlin 1989.

[6] *S. Eilenberg:* Automata, Languages, and Machines,Volume A. Academic Press. New York 1974.

[7] *H. Fernau:* Membership for 1-Limited ET0L Languages Is Not Decidable. J. Inform. Process. Cybernet. EIK **30**(1994), 191–211.

[8] *M. Frings:* Systeme mit eingeschränkt paralleler Ersetzung. Diplomarbeit, Technische Universität Braunschweig, 1985.

[9] *S. Ginsburg:* The mathematical theory of context-free languages. McGraw-Hill, New York 1966.

[10] *S. Ginsburg:* Algebraic and automata-theoretic properties of formal language. North-Holland, Amsterdam 1975.

[11] *M.A. Harrison:* Introduction to Formal Language Theory. Addison-Wesley, Reading 1978.

[12] *G.T. Herman, G. Rozenberg:* Developmental Systems and Languages. North-Holland, 1975.

[13] *H. Hermes:* Aufzählbarkeit, Entscheidbarkeit, Berechenbarkeit. Springer, Berlin 1961.

[14] *W.M.L. Holcombe:* Algebraic automata theory. Cambridge University Press, Cambridge 1982.

[15] *J.E. Hopcroft, J.D. Ullman:* Introduction to automata theory, languages, and computation, Addison-Wesley, Reading 1979.

[16] *G. Hotz, Th. Kretschmer:* The Power of the Greibach Normal Form. J. Inf. Process. Cybernet. **EIK 25**(1989), 507–512.

[17] *R.N. Moll, M.A. Arbib, A.J. Kfoury:* An Introduction to Formal Language Theory. Springer, New York 1988.

[18] *M. Penttonen:* One-Sided and Two-Sided Context in Formal Grammars. Information and Control **25**(1974), 371–392.

[19] *P. Prusienkiewicz, J. Hanan:* Lindenmayer Systems, Fractals, and Plants. Lecture Notes in Biomathematics, Vol **79**. Springer. Berlin 1989.

[20] *P. Prusienkiewicz, A. Lindenmayer:* The Algorithmic Beauty of Plants. Springer, Berlin 1990.

[21] *G.E. Révész:* Introduction to Formal Languages. McGraw-Hill, New York 1983.

[22] *G. Rozenberg, A. Salomaa:* The mathematical theory of L Systems. Academic Press, New York 1980.

[23] *A. Salomaa:* Formal Languages. Academic Press, New York 1973.

[24] *A. Salomaa:* Formale Sprachen. Springer, Berlin 1978.

[25] *A. Salomaa:* Computation and Automata. Cambridge University Press, Cambridge 1985.

[26] *P.H. Starke:* Abstrakte Automaten. Deutscher Verlag der Wissenschaften, Berlin 1969.

[27] *R. Szelepcsényi:* The Method of Forcing for Nondeterministic Automata. EATCS **33**(1987), 96–100.

[28] *D. Wätjen:* k-limited 0L Systems and Languages. J. Inf. Process. Cybernet. EIK **24**(1988), 267–285.

[29] *D. Wätjen:* On k-uniformly-limited T0L Systems and Languages. J. Inf. Process. Cybernet. EIK **26**(1990), 229–238.

[30] *D. Wätjen:* Theoretische Informatik. Eine Einführung. Oldenbourg Verlag, München 1994.

Index