Formale Sprachen und Automaten

Vorlesung WS 1997/98

Universität München
Centrum für Informations-
und Sprachverarbeitung

Hans Leiß

20. Dezember 1997

Inhaltsverzeichnis

Vorwort

Die Theorie der formalen Sprachen und der Automaten, die man zur Erkennung und Syntaxanalyse von Ausdrücken solcher Sprachen braucht, wurde in der Linguistik durch Chomskys Grammatiktheorien der fünfziger Jahre begründet und in der Informatik bis etwa Mitte der siebziger Jahre weiterentwickelt.

Einige grundsätzliche Fragen wurden aber erstaunlicherweise erst 1990-1992 geklärt. Ein Ziel dieser Vorlesung war es, diese neuen Ergebnisse von Kozen und Pratt zur endlichen Axiomatisierbarkeit der Gleichungstheorie regulärer Sprachen wenigstens vorzustellen.

Ein weiteres Ziel war, den Zusammenhang zwischen Programmierung und Sprachanalyse deutlicher als üblich hervorzuheben, um Studenten der Computerlinguistik einen Einblick in die Theorie der Berechenbarkeit zu vermitteln. Dazu wurden reguläre Operationen um einen Fixpunktoperator erweitert, so daß man den Zusammenhang zwischen regulären Sprachen und iterativen sequentiellen Programmen einerseits und zwischen kontextfreien Sprachen und rekursiven Programmen andererseits klar erkennt.

Im vorletzten Kapitel wird auf einige Grammatikformalismen der Computerlinguistik eingegangen, die bisher nur zur Beschreibung von natürlichen Sprachen verwendet werden. Hier stellen wir mit den Baumadjunktionsgrammatiken eine Möglichkeit dar, eine Klasse von "schwach kontextsensitiven" Sprachen zu definieren. Außerdem zeigen wir ein frühes Ergebnis von Gaifman über die Äquivalenz von Kategorialgrammatiken und kontextfreien Grammatiken, sowie den erst 1992 bewiesenen Satz von Pentus, daß auch Lambek-Grammatiken genau die kontextfreien Sprachen definieren.

Aus Zeitgründen konnten weitere in der Computerlinguistik übliche Grammatikformalismen und ihr Zusammenhang zur klassischen Theorie der formalen Sprachen nicht mehr behandelt werden. Naheliegend wäre es, die Projektionstheorie der Phrasen in Form der $\bar{X}$-Grammatiken und den Kopfbegriff in Form der Kopfgrammatiken nach Pollard als weitere linguistisch motivierte Varianten von kontextfreien Grammatiken zu untersuchen. Ein zusätzliches wichtiges Thema wäre, den Übergang von der Gleichungsdefinierbarkeit kontextfreier Sprachen zu induktiven Sprachdefinitionen durch Hornformeln (Definite Clause Grammars) darzustellen. Dies würde die Theorie der formalen Sprachen besser mit der Praxis der Computerlinguistik verbinden.

Überhaupt scheint es lohnend, statt der Ableitung in einer Phrasenstrukturgrammatik den Begriff der logischen Definierbarkeit formaler Sprachen stärker zu betonen. Bekannte Zusammenhänge zwischen Komplexitätstheorie und endlicher Modelltheorie ergäben über die Darstellung von Wörtern als endliche Strukturen elegante Beweise über die Erkennungskomplexität natürlich definierter Sprachklassen. Leider konnte das aus Zeitgründen nicht mehr ausgearbeitet werden.

H.Leiß

1 Mathematische Grundbegriffe

1.1 Relationen und Funktionen

Seien $A_1, \ldots, A_n$ Mengen. Ein n-Tupel über $A_1, \ldots, A_n$ ist eine Folge $(a_1, \ldots, a_n)$ von Objekten $a_i \in A_i$, für $i = 1, \ldots, n$. Zwei n-Tupel sind gleich, falls sie dieselben Elemente in derselben Reihenfolge enthalten

$$(a_1, \ldots, a_n) = (b_1, \ldots, b_n) \iff a_i = b_i \text{ für } i = 1, \ldots, n.$$

Das cartesische Produkt der Mengen $A_1, \ldots, A_n$ ist

$$A_1 \times \ldots \times A_n := \{ (a_1, \ldots, a_n) \mid a_i \in A_i \text{ für } i = 1, \ldots n \}.$$

Eine n-stellige Relation auf $A_1, \ldots, A_n$ ist eine Menge $R \subseteq A_1 \times \ldots \times A_n$ von n-Tupeln. Eine partielle Funktion von $A_1 \times \ldots \times A_n$ nach B ist eine Relation $R \subseteq A_1 \times \ldots \times A_n \times B$ mit

$$(a_1, \ldots, a_n, b) \in R, \quad (a_1, \ldots, a_n, c) \in R \implies b = c.$$

Die Funktion R heißt total, falls es zu jedem $(a_1, \ldots, a_n)$ ein $b \in B$ mit $(a_1, \ldots, a_n, b) \in R$ gibt. Für Funktionen $f \subseteq A_1 \times \ldots \times A_n \times B$ schreiben wir ab jetzt

$$f : A_1 \times \ldots \times A_n \longrightarrow B,$$

und $f(a_1, \ldots, a_n) = b$ statt $(a_1, \ldots, a_n, b) \in f$. Das Bild von $R \subseteq A_1 \times \ldots \times A_n$ unter $f : A_1 \times \ldots \times A_n \to A$ ist die Menge

$$\{ f(a_1, \ldots, a_n) \mid (a_1, \ldots, a_n) \in R \} =: f[R].$$

Eine n-stellige Relation auf A ist eine Relation $R \subseteq A^n := A \times \ldots \times A$ mit n gleichen Faktoren A.

Eine Teilmenge $B \subseteq A$ ist abgeschlossen unter der Relation $R \subseteq A^n \times A$, falls

$$(b_1, \ldots b_n, a) \in R,\ b_i \in B \Longrightarrow a \in B.$$

Beispiel 1.1 Sei $A = A_1 \times \ldots \times A_n$, und $2^A := \{ S \mid S \subseteq A \}$ die Potenzmenge von A. Sie ist abgeschlossen unter den boolschen Operationen $\cap, \cup, \overline{}$:

$$\begin{array}{lcl} R_1 \cap R_2 & := & \{ (a_1, \ldots, a_n) \mid (a_1, \ldots, a_n) \in R_1 \text{ und } (a_1, \ldots, a_n) \in R_2 \}, \\ R_1 \cup R_2 & := & \{ (a_1, \ldots, a_n) \mid (a_1, \ldots, a_n) \in R_1 \text{ oder } (a_1, \ldots, a_n) \in R_2 \}, \\ \overline{R} & := & \{ (a_1, \ldots, a_n) \mid (a_1, \ldots, a_n) \in A, (a_1, \ldots, a_n) \notin R \}. \end{array}$$

Sei $R \in 2^A$ eine n-stellige Relation und $\mathcal{R} = \{ S \subseteq A^n \mid S \subseteq R \}$ die Gesamtheit aller Teilrelationen von R. Dann ist $\mathcal{R}$ abgeschlossen unter der Relation $\subseteq$ zwischen Relationen und unter den Operationen $\cap$ und $\cup$, aber nicht unter der Komplementbildung.

Sei $f : A^n \to A$ und $B \subseteq A$. Der Abschluss von B unter f ist

$$f^\infty(B) := \bigcap\{\, S \mid B \subseteq S \subseteq A,\ \forall a_1, \ldots, a_n \in S\ f(a_1, \ldots, a_n) \in S \,\},$$

d.h. die kleinste Teilmenge von A, die B umfaßt und unter f abgeschlossen ist.

Lemma 1.2 Sei $f : A^n \to A$ und $B \subseteq A$. Der Abschluß von B unter f ist

$$\bigcup\{\, B_i \mid i \in I\!N \,\}, \qquad \text{mit } B_0 := B,\ B_{i+1} := B_i \cup \{\, f(a_1, \ldots, a_n) \mid (a_1, \ldots, a_n) \in B_i^n \,\}.$$

Beweis: $f^\infty(B) \subseteq \bigcup\{\, B_i \mid i \in I\!N \,\}$: Es genügt zu zeigen, daß $\bigcup\{\, B_i \mid i \in I\!N \,\}$ unter f abgeschlossen ist. Seien $a_1, \ldots, a_n \in \bigcup\{\, B_i \mid i \in I\!N \,\}$. Es gibt ein kleinstes $j \in I\!N$ mit $\{a_1, \ldots, a_n\} \subseteq B_j$. Dann ist $f(a_1, \ldots, a_n) \in B_{j+1} \subseteq \bigcup\{\, B_i \mid i \in I\!N \,\}$.

$\bigcup\{\, B_i \mid i \in I\!N \,\} \subseteq f^\infty(B)$: Offenbar ist $B_0 = B \subseteq f^\infty(B)$. Ist für ein $i \in I\!N$ schon $B_i \subseteq f^\infty(B)$, so ist auch $B_{i+1} \subseteq f^\infty(B)$, da diese Menge unter f abgeschlossen ist. □

1.2 Die Algebra der zweistelligen Relationen

Definiere zu zweistelligen Relationen R, R_1, R_2 auf einer Menge A: Die Inverse von R:

$$R^{-1} := \{\, (b, a) \mid (a, b) \in R \,\},$$

das Relationenprodukt von R_1 und R_2

$$R_1 \circ R_2 := \{\, (a, c) \mid \exists\, b \in A\, ((a, b) \in R_1 \wedge (b, c) \in R_2) \,\},$$

die Identität auf A:

$$1_A = Id_A := \{\, (a, a) \mid a \in A \,\},$$

die Relationensumme von R_1 und R_2:

$$R_1 \cup R_2 := \{\, (a, b) \mid (a, b) \in R_1 \text{ oder } (a, b) \in R_2, \,\}$$

und die leere Relation auf A:

$$0_A := \emptyset,$$

d.h. die Relation, die kein Tupel enthält.

Beachte, daß damit gilt:

$$\begin{array}{ccccccc}
R \cup (R_1 \cup R_2) & = & (R \cup R_1) \cup R_2, & \qquad R \cup 0 & = & R & = 0 \cup R, \\
R_1 \cup R_2 & = & R_2 \cup R_1, & & & & \\
R \cup R & = & R, & & & & \\
R \circ (R_1 \circ R_2) & = & (R \circ R_1) \circ R_2, & \qquad R \circ 1 & = & R & = 1 \circ R, \\
(R_1 \cup R_2)^{-1} & = & R_1^{-1} \cup R_2^{-1}, & \qquad R \circ 0 & = & 0 & = 0 \circ R, \\
(R_1 \circ R_2)^{-1} & = & R_2^{-1} \circ R_1^{-1}, & & 1^{-1} & = & 1, \\
(R^{-1})^{-1} & = & R, & & 0^{-1} & = & 0.
\end{array}$$

Aber i.a. gilt *nicht* etwa $R_1 \circ R_2 = R_2 \circ R_1$ oder $R \circ R^{-1} = 1_A$.

Beispiel 1.3 Sei $R = \{(a,b),(c,b)\}$, mit paarweise verschiedenen Punkten a, b und c. Dann ist $(a,c) \in R \circ R^{-1}$, da ja $(a,b) \in R$ und $(b,c) \in R^{-1}$. Wegen $a \neq c$ ist aber $(a,c) \notin 1_A$, also $R \circ R^{-1} \not\subseteq 1_A$.

Eine Relation R heißt TRANSITIV, falls $R \circ R \subseteq R$, d.h. für alle a,b,c gilt: $(a,b) \in R, (b,c) \in R \Longrightarrow (a,c) \in R$. R heißt REFLEXIV, falls $Id \subseteq R$, d.h. $(a,a) \in R$ für alle $a \in A$, und SYMMETRISCH, falls $R^{-1} \subseteq R$, d.h. $(a,b) \in R \Longrightarrow (b,a) \in R$.

Lemma 1.4 Ist $\{ R_i \mid i \in I \}$ eine Familie von reflexiven (transitiven, symmetrischen) Relationen auf A, so ist auch $\bigcap_{i \in I} R_i := \{ (a,b) \mid (a,b) \in R_i \text{ für alle } i \in I \}$ eine reflexive (bzw. transitive, symmetrische) Relation.

Beweis: Wegen $1_A \subseteq R_i$ für alle $i \in I$ gilt auch $1_A \subseteq \bigcap_{i \in I} R_i$. Analog: Ist $R_i \circ R_i \subseteq R_i$, für alle $i \in I$, so ist

$$\left(\bigcap_{i \in I} R_i\right) \circ \left(\bigcap_{j \in I} R_j\right) \subseteq R_i \circ R_i \subseteq R_i$$

für alle i, also auch $\subseteq \bigcap_{i \in I} R_i$. Entsprechend für die Symmetrie. □

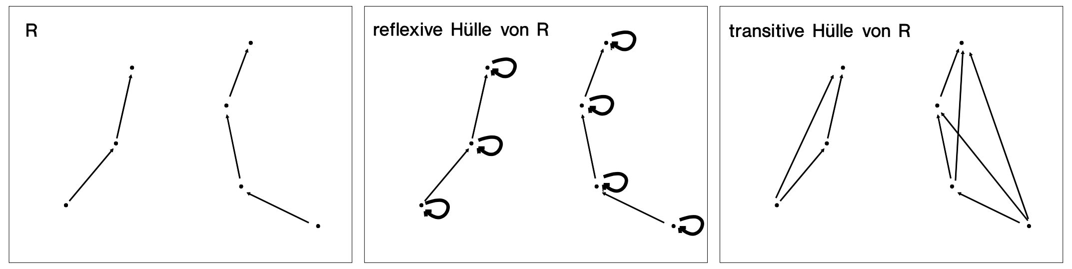
Daher definieren wir für $R \subseteq A \times A$ den TRANSITIVEN ABSCHLUSS VON R,

$$R^+ := \bigcap \{ S \mid R \subseteq S \subseteq A \times A, \quad S \text{ ist transitiv} \},$$

und den REFLEXIVEN UND TRANSITIVEN ABSCHLUSS VON R,

$$R^* := \bigcap \{ S \mid R \subseteq S \subseteq A \times A, \quad S \text{ ist transitiv und reflexiv} \}.$$

R^+ (bzw. R^*) ist die $\subseteq$-kleinste transitive (bzw. kleinste transitive und reflexive) Relation S auf A ist, die R umfaßt.



Ähnlich wie in Lemma 1.2 zeigt man:

$$R^* = \bigcup \{ R^n \mid n \in I\!N \}, \qquad \text{mit } R^0 := 1_M,\ R^{n+1} := R^n \circ R.$$

Eine ALGEBRA $\mathcal{A} = (A, f_1, \ldots, f_k)$ besteht aus einer Trägermenge A und einer Reihe $f_1, \ldots, f_k$ von Operationen auf A, d.h. Abbildungen $f_i : A^{n_i} \longrightarrow A$.

Die Algebra $\mathcal{R}_A := (2^{A \times A}, \cup, \emptyset, \circ, 1_A, {}^{-1}, {}^*)$ heißt DIE ALGEBRA DER 2-STELLIGEN RELATIONEN AUF A.

1.3 Äquivalenzrelationen

Eine zweistellige Relation R auf einer Menge A heißt eine Äquivalenzrelation auf A, falls R reflexiv, transitiv und symmetrisch ist. Ist R eine Äquivalenzrelation auf A, so sei

$$[a]_R := \{\, b \mid b \in A,\ (a,b) \in R \,\}$$

die Äquivalenzklasse von a bezüglich R. Die Gesamtheit aller Äquivalenzklassen von R wird mit A/R bezeichnet.

Eine Mengenfamilie $\mathcal{B} \subseteq 2^A$ ist eine Zerlegung von A in disjunkte Teilmengen, wenn es für jedes $a \in A$ genau ein $B \in \mathcal{B}$ mit $a \in B$ gibt.

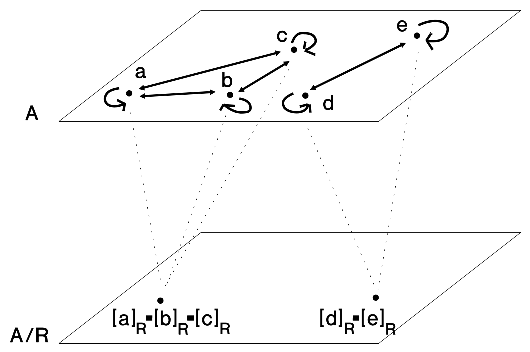
Satz 1.5 (i) Ist R eine Äquivalenzrelation auf A, so ist $\{\, [a]_R \mid a \in A \,\}$ eine Zerlegung von A in disjunkte Teilmengen.

(ii) Ist $\mathcal{B} \subseteq 2^A$ eine Zerlegung von A in disjunkte Teilmengen, so wird durch

$$(a,b) \in R :\Longleftrightarrow \{a,b\} \subseteq B \text{ für ein } B \in \mathcal{B}$$

eine Äquivalenzrelation auf A definiert.

Zur Veranschaulichung betrachte man folgendes Bild:



Beweis:

(i) Beachte, daß $a \in [a]_R$. Also bilden die $[a]_R$ eine Zerlegung von A. Es bleibt zu zeigen, daß verschiedene Teilmengen $[a]_R$ zueinander disjunkt sind.

Seien $a, b \in A$ mit $[a]_R \cap [b]_R \neq \emptyset$, etwa $c \in [a]_R \cap [b]_R$. Für jedes $x \in A$ gilt:

$$\begin{array}{rcll} x \in [a]_R & \Longrightarrow & (a,x) \in R, & \text{nach der Definition von } [a]_R, \\ & \Longrightarrow & (x,a) \in R, & \text{wegen der Symmetrie von } R, \\ & \Longrightarrow & (x,c) \in R, & \text{mit } (a,c) \in R \text{ und der Transitivität von } R, \\ & \Longrightarrow & (c,x) \in R, & \text{wegen der Symmetrie von } R, \\ & \Longrightarrow & (b,x) \in R, & \text{mit } (b,c) \in R \text{ und der Transitivität von } R, \\ & \Longrightarrow & x \in [b]_R. & \end{array}$$

Daher ist $[a]_R \subseteq [b]_R$, und analog gilt auch $[a]_R \supseteq [b]_R$. Also sind zwei sich überlappende Mengen $[a]_R$ und $[b]_R$ schon gleich, d.h. die Zerlegung ist disjunkt.

(ii) Das bleibt dem Leser zur Übung überlassen.

□

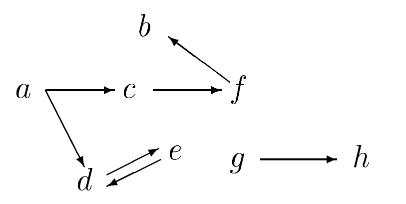
Beispiel 1.6 Sei $f : A \to B$ eine totale Funktion. Dann definiert $(a,b) \in R :\Longleftrightarrow f(a) = f(b)$ eine Äquivalenzrelation auf A.

1.4 Partielle Ordnungen, Graphen, Bäume

Ein GRAPH $\mathcal{G} = (A, R)$ ist eine Menge A mit einer 2-stelligen Relation R auf A. Falls R symmetrisch ist, heißt G UNGERICHTET. Ein WEG VON a NACH b in $\mathcal{G}$ ist eine Folge $a = a_1, \ldots, a_n = b$ von Elementen aus A mit $(a_i, a_{i+1}) \in R$ für $i = 1, \ldots, n-1$.

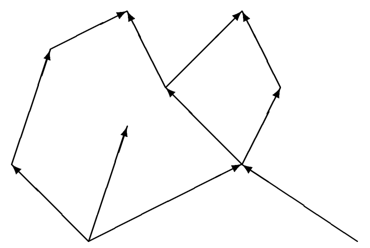
Man schreibt meist $a \to b$ statt $(a,b) \in R$.

Beispiel 1.7 Ein Graph $(A, \to)$ mit einem Weg von a nach b:

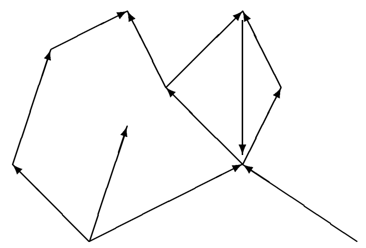


Eine PARTIELLE ORDNUNG (A, R) ist eine Menge A mit einer reflexiven, transitiven und antisymmetrischen Relation R auf A. Eine Relation $R \subseteq A \times A$ heißt ANTISYMMETRISCH, falls $R \cap R^{-1} \subseteq Id_A$.

Beispiel 1.8 Zwei Graphen $(A, \to)$, für die $\to^*$ antisymmetrisch bzw. nicht antisymmetrisch ist:



$\to^*$ ist eine partielle Ordnung



$\to^*$ ist keine partielle Ordnung

Eine partielle Ordnung schreibt man meist als $(A, \leq)$. Man sagt a ist ECHT KLEINER ALS b und schreibt $a < b$, falls $a \leq b$ und $a \neq b$.

Eine partielle Ordnung $\leq$ auf A ist TOTAL, falls $a \leq b$ oder $b \leq a$ für alle $a, b \in A$ gilt, d.h. wenn je zwei Elemente vergleichbar sind.

Eine partielle Ordnung $(A, \leq)$ ist FUNDIERT, falls es keine unendliche echt absteigende Folge

$$\ldots < a_{n+1} < a_n < \ldots < a_1 < a_0 \qquad n \in I\!N,\ a_n \in A,$$

gibt.

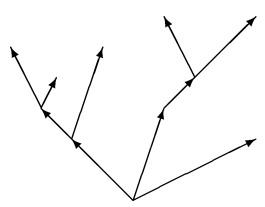
Ein (nicht angeordneter) BAUM ist eine partielle Ordnung $(B, \leq)$ mit den Eigenschaften

(i) $(B, \leq)$ ist fundiert.

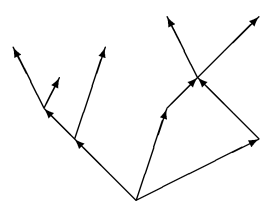
(ii) Für jedes $b \in B$ ist $(\{\, a \in B \mid a \leq b \,\}, \leq)$ eine totale Ordnung.

(iii) Es gibt ein kleinstes Element $0 \in B$, die WURZEL, mit $0 \leq b$ für alle $b \in B$.

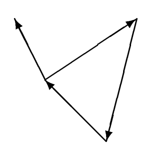
Beispiel 1.9 Drei Graphen $(A, \to)$, bei denen $\to^*$ ein Baum bzw. kein Baum ist:



$\to^*$ ist ein Baum

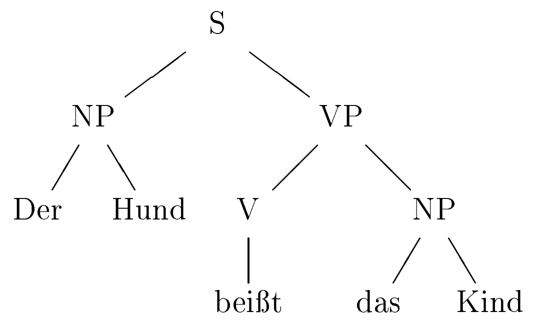


$\to^*$ ist kein Baum



$\to^*$ ist kein Baum

Wir werden nur endliche Bäume brauchen, und zwar zur Darstellung von syntaktischen Strukturen:



Dazu werden die Punkte (Knoten) des Baums mit Symbolen markiert, die die üblichen syntaktischen Kategorien (Nominalphrase, Verbalphrase usw.) repräsentieren.

Bei diesen Bäumen kommt es auch auf eine Anordnung der Zweige an, was in der bisherigen Definition nicht berücksichtigt ist.

1.5 Alphabete und Formale Sprachen

Eine natürliche Sprache kann man sehr grob als eine Menge von Wort- (bzw. Buchstaben- oder Laut-) Folgen auffassen. Diese Abstraktion erweist sich als nützlich, obwohl dabei von wesentlichen Eigenschaften natürlicher Sprachen abgesehen wird.

Definition 1.10 Sei Σ eine endliche Menge, das ALPHABET. Ein WORT w über Σ ist eine endliche Folge $(a_1, \dots, a_n)$ von Buchstaben aus Σ, kurz $w = a_1 \dots a_n$. Mit

$$\Sigma^* := \{ \, w \mid w \text{ ist ein Wort über } \Sigma \, \}$$

wird die Menge aller Wörter über Σ bezeichnet. Das LEERE WORT ist die Folge () ohne Elemente, oft mit ε bezeichnet. Sind w, v Wörter, so ist auch ihre VERKETTUNG $w \cdot v$ wieder ein Wort:

$$(a_1, \dots, a_n) \cdot (b_1, \dots, b_m) := (a_1, \dots, a_n, b_1, \dots, b_m)$$

Wir betrachten $a \in \Sigma$ auch als das Wort (a) über Σ. Meistens schreiben wir vw statt $v \cdot w$. Die LÄNGE $|w|$ von w ist gegeben durch $|\varepsilon| = 0$ und $|wa| = |w| + 1$ für $a \in \Sigma$.

Beispiel 1.11 (i) Für Wörter $v, w \in \Sigma^*$ sei $v \leq_{Länge} w \;:\iff\; |v| \leq |w|$, d.h. wenn v höchstens so lang wie w ist. Die Relation $\leq_{Länge}$ ist keine partielle Ordnung auf Σ^*, da sie nicht antisymmetrisch ist: zum Beispiel ist *kurz* $\leq_{Länge}$ *lang* und *lang* $\leq_{Länge}$ *kurz*, aber diese Wörter sind verschieden.

(ii) Die Teilwortrelation, definiert durch

$$u \leq_{infix} w :\Longleftrightarrow \exists v_1 \in \Sigma^* \, \exists v_2 \in \Sigma^* \; v_1 u v_2 = w,$$

ist eine partielle Ordnung auf Σ^*, die nicht total, aber fundiert ist. Das gleiche gilt für die Präfix- und Suffixrelationen, die durch

$$\begin{aligned} u \leq_{pr\ddot{a}fix} w \; &:\Longleftrightarrow \; \exists v \in \Sigma^* \; uv = w, \qquad \text{und} \\ u \leq_{suffix} w \; &:\Longleftrightarrow \; \exists v \in \Sigma^* \; vu = w \end{aligned}$$

definiert sind.

(iii) Sei Σ mit einer vorgegebenen totalen Ordnung $\leq_\Sigma$ der Buchstaben versehen. Die lexikographische Ordnung $v \leq_{lex} w$ von Wörtern $v, w \in \Sigma^*$ ist wie folgt definiert:

$$\begin{aligned} v \leq_{lex} w \; &:\Longleftrightarrow \; v = w \text{ oder } v <_{lex} w, \\ a_1 \cdots a_n <_{lex} b_1 \cdots b_m \; &:\Longleftrightarrow \\ (i) \quad & n = 0 < m \qquad \text{oder} \\ (ii) \quad & n \neq 0 \neq m \;\wedge\; a_1 <_\Sigma b_1 \qquad \text{oder} \\ (iii) \quad & n \neq 0 \neq m \;\wedge\; a_1 = b_1 \;\wedge\; a_2 \cdots a_n <_{lex} b_2 \cdots b_m. \end{aligned}$$

Die lexikographische Ordnung $(\Sigma^*, \leq_{lex})$ ist total, aber nicht fundiert: sind $a <_\Sigma b$ aus Σ, so bildet

$$a^{n+1} b \; <_{lex} \; a^n b, \qquad n \in I\!N,$$

eine unendliche echt absteigende Folge. Hier ist $a^0 := \varepsilon$ und $a^{n+1} := a^n \cdot a$.

Proposition 1.12 $(\Sigma^*, \cdot, \varepsilon)$ ist ein Monoid oder eine freie Halbgruppe, d.h. folgende Bedingungen sind erfüllt:

(i) $\cdot$ ist assoziativ: $\forall u, v, w \in \Sigma^* : \; (u \cdot (v \cdot w)) = ((u \cdot v) \cdot w)$

(ii) ε ist neutrales Element bzgl. $\cdot$: $\forall w \in \Sigma^* : \; \varepsilon \cdot w = w \cdot \varepsilon = w$

(iii) $\forall a_1, \ldots, a_n, b_1, \ldots b_m \in \Sigma : \; a_1 \ldots a_n = b_1 \ldots b_m \Longrightarrow n = m \wedge a_1 = b_1 \wedge \ldots \wedge a_n = b_n$.

Definition 1.13 Eine formale Sprache über Σ ist eine Wortmenge $L \subseteq \Sigma^*$.

Da formale Sprachen beliebige Teilmengen von Σ^* sind, ist die Gesamtheit aller formalen Sprachen über Σ abgeschlossen unter den mengentheoretischen Operationen:

$$\overline{L} = \Sigma^* - L, \quad L_1 \cup L_2, \quad L_1 \cap L_2, \quad \emptyset, \quad \Sigma^*.$$

Hauptsächlich werden wir folgende Operationen betrachten, unter denen die Klasse der Sprachen ebenfalls abgeschlossen ist:

$$\begin{aligned}
L_1 \cdot L_2 &:= \{\, w_1 \cdot w_2 \mid w_1 \in L_1, w_2 \in L_2 \,\} \\
L^* &:= \bigcap \{\, S \mid L \subseteq S \subseteq \Sigma^*,\ \varepsilon \in S,\ S \cdot S \subseteq S \,\} \\
L \backslash M &:= \{\, w \mid \text{ für alle } v \in L \text{ ist } v \cdot w \in M \,\} \\
M / L &:= \{\, w \mid \text{ für alle } v \in L \text{ ist } w \cdot v \in M \,\}
\end{aligned}$$

Dabei heißen $L_1 \cdot L_2$ das Sprachprodukt von L_1 und L_2, $L\backslash M$ die Linksdivsion von M durch L und M/L die Rechtsdivision von M durch L.

Beachte, daß zum Beispiel $L \cdot (L\backslash M) \subseteq M$ und $(M/L) \cdot L \subseteq M$ gelten. Die Sprachdivisionen \ und / spielen eine wesentliche Rolle bei der Beschreibung von Sprachen durch Kategorialgrammatiken (vgl. Kapitel 10).

L^* ist der Abschluss von L unter Verkettung und ε, oder kurz die Sprachiteration von L, und kann auch so dargestellt werden:

$$\begin{aligned}
L^* &= \bigcup \{\, L^n \mid n \in I\!N \,\}, \qquad \text{mit } L^0 := \{\varepsilon\},\ L^{n+1} := L^n \cdot L, \\
&= \{\, w_1 \cdots w_n \mid n \geq 1,\ w_1 \in L, \ldots, w_n \in L \,\} \cup \{\varepsilon\}
\end{aligned}$$

Gelegentlich braucht man auch die Spiegelung L^{-1} einer Sprache L, die durch

$$L^{-1} := \{\, w^{-1} \mid w \in L \,\} \qquad \text{mit } \varepsilon^{-1} := \varepsilon,\ (w \cdot a)^{-1} := a \cdot w^{-1}$$

definiert ist. Sie verhält sich zur Vereinigung und zum Sprachprodukt ähnlich wie die Inverse zur Vereinigung und zum Relationenprodukt, d.h. es gilt

$$(L_1 \cup L_2)^{-1} = L_1^{-1} \cup L_2^{-1}, \qquad (L_1 \cdot L_2)^{-1} = L_2^{-1} \cdot L_1^{-1}, \qquad (L^{-1})^{-1} = L.$$

Beispiel 1.14 Sei $\Sigma = \{a, b, \ldots, z\}$. Dann ist

(i) $\{a, b\} \cdot \Sigma^*$ = Menge aller Wörter, die mit a oder b beginnen,

(ii) $\Sigma^* w \Sigma^*$ = Menge aller Wörter, in denen das Teilwort w vorkommt.

(iii) Ist $N \supseteq$ {Haus, Tor, Schloß, Schlüssel, ...} die Menge der einfachen deutschen Nomina, so umfaßt N^* die Nomenkomposita, z.B. Haustor, Torschloß, Haustorschlüssel.

(iv) $P := \{\, w \cdot w^{-1} \mid w \in \Sigma^* \,\}$ heißt die Sprache der Palindrome über Σ.

Palindrome sind also Wörter oder Sätze, die sich vorwärts und rückwärts lesen lassen, z.B. Retter, Otto, neppen. Ein schönes Kunststück ist folgendes Palindromgedicht von H. Pfeiffer[18]:

…'s nutzt uns!

Eins nutzt uns:
Amore!

Die Rederei da,
die Rederei der Omas,
nutzt uns nie.

Aufgaben

Aufgabe 1.1 Eine 2-stellige Funktion $+ : A \times A \to A$ heißt (a) IDEMPOTENT, wenn $a + a = a$ für alle $a \in A$, (b) ASSOZIATIV, wenn $a + (b + c) = (a + b) + c$ für alle a, b und $c \in A$, und (c) KOMMUTATIV, wenn $a + b = b + a$ für alle a und $b \in A$. Definiere auf A eine zweistellige Relation $\leq$ durch

$$b \leq c \quad :\Longleftrightarrow \quad b + c = c.$$

Man zeige:

(i) $\leq$ ist reflexiv genau dann, wenn $+$ idempotent ist,

(ii) $\leq$ ist transitiv, wenn $+$ assoziativ ist,

(iii) $\leq$ ist antisymmetrisch, wenn $+$ kommutativ ist.

Wenn also $+$ idempotent, assoziativ, und kommutativ ist, so ist $\leq$ eine partielle Ordnung.

Bem. Ist $+$ die Vereinigung von Mengen, so ist $\leq$ die Teilmengenbeziehung $\subseteq$.

Aufgabe 1.2 Sei $A = \{a, b, c, d\}$ und $R = \{(a, b), (a, c), (a, d), (b, d), (d, c)\}$. Man gebe den reflexiven und den transitiven Abschluß von R an, und den reflexiven und transitiven Abschluß R^*. (Bilder zeichnen!)

Aufgabe 1.3 Sei $(A, \leq^A)$ eine partiell geordnete Menge, und $B \subseteq A$. Definiere

$$\leq^B := \{(b, c) \mid b, c \in B, b \leq^A c\}.$$

Man zeige, daß auch $(B \leq^B)$ eine partiell geordnete Menge ist.

Aufgabe 1.4 Seien $\{R_i \mid i \in I\}$ und $\{S_j \mid j \in J\}$ beliebige Familien von zweistelligen Relationen auf der Menge M. Man zeige, daß

$$\bigcup\{R_i \circ S_j \mid i \in I, j \in J\} = \bigcup\{R_i \mid i \in I\} \circ \bigcup\{S_j \mid j \in J\}.$$

Aufgabe 1.5 Sei $A_1 = \{abb, aab, aba\}$ und $A_2 = \{\, (abaa)^n \mid 1 \leq n \leq 10 \,\}$. Berechne die Bedeutung $L(r)[A_1, A_2]$ fr den regulären Ausdruck $r = (aX_1)^* + X_2X_1$

2 Reguläre Ausdrücke

Wir wollen (i.a. unendliche) Sprachen mit endlichen Mitteln darstellen, z.B. durch Grammatiken, nach denen die Sätze der Sprache gebildet werden dürfen. Es ist sinnvoll, von der Wahl des jeweiligen Alphabets Σ oder konkreten Sprachen L_0, L_1, L_2 zu abstrahieren, wenn es um die Gültigkeit gewisser Gesetze wie

$$L_0 \cdot (L_1 \cup L_2) = L_0 \cdot L_1 \cup L_0 \cdot L_2$$

geht. Wir beschäftigen uns jetzt mit Möglichkeiten, Sprachen zu konstruieren. Dabei beschränken wir uns auf einige der schon betrachteten Operationen:

Definition 2.1 Reguläre Ausdrücke über Σ sind die folgenden und nur diese:

(i) 0 und 1 sind reguläre Ausdrücke über Σ.

(ii) Für jedes $a \in \Sigma$ ist a ein regulärer Ausdruck über Σ.

(iii) Sind r und s reguläre Ausdrücke über Σ, so auch $(r \cdot s)$, $(r + s)$, und r^*.

Statt $(r \cdot s)$ schreiben wir in Zukunft meistens rs, gelegentlich auch $(r; s)$.

In den folgenden Abschnitten behandeln wir drei verschiedene natürliche Interpretationen für reguläre Ausdrücke: Sprachen, zweistellige Relationen und iterative sequentielle Programme.

Für das Komplement und die Inverse R^{-1} einer Relation R bzw. die Spielgelung L^{-1} einer Sprache L wurde kein reguärer Ausdruck eingeführt. Der Grund liegt darin, daß diese Operationen bei der Interpretation als Programme keinen Sinn machen. Dasselbe gilt für die Divisionen \ und /, auf die wir aber in Abschnitt 3.4 zurückkommen.

2.1 Reguläre Ausdrücke als formale Sprachen

Sei $(\Sigma^*, \cdot, \varepsilon)$ das Monoid der Wörter über Σ. Dann können wir jeden regulären Ausdruck r über Σ durch eine Sprache $L(r)$ interpretieren:

$$\begin{array}{lll@{\qquad}lll}
L(0) & := & \emptyset & L(r+s) & := & L(r) \cup L(s) \\
L(1) & := & \{\varepsilon\} & L(r \cdot s) & := & L(r) \cdot L(s) \\
L(a) & := & \{a\}, \quad \text{für } a \in \Sigma & L(r^*) & := & L(r)^*.
\end{array}$$

Mit anderen Worten, wir interpretieren reguläre Ausdrücke als formale Sprachen in der ALGEBRA ALLER SPRACHEN ÜBER Σ,

$$\mathcal{L}_\Sigma := (2^{\Sigma^*}, \cup, \emptyset, \cdot, \{\varepsilon\}, ^*, \{a\})_{a\in\Sigma}.$$

Verschiedene Ausdrücke können dieselbe Sprache bezeichnen, z.B.

$$L(r) = L(r+0), \qquad L(0^*) = L(1), \qquad L(r+s) = L(s+r), \qquad L(r\cdot 0) = L(0).$$

Sprachen lassen sich natürlich auch anders als durch reguläre Ausdrücke beschreiben, etwa durch eine charakteristische Eigenschaft:

$$\begin{aligned} L((a+b)^*a) &= L((a+b)^*)\cdot L(a) \\ &= (L(a)\cup L(b))^*\cdot L(a) \\ &= (\{a\}\cup\{b\})^*\cdot L(a) = \{a,b\}^*\{a\} \\ &= \{w\in\{a,b\}^* \mid w\neq\varepsilon \text{ und der letzte Buchstabe von } w \text{ ist } a\} \end{aligned}$$

Definition 2.2 Die Klasse Reg_Σ der REGULÄREN SPRACHEN ÜBER Σ ist die kleinste Klasse $\mathcal{L}$ mit

(i) $\emptyset\in\mathcal{L}$ und $\{\varepsilon\}\in\mathcal{L}$.

(ii) $\{a\}\in\mathcal{L}$ für jedes $a\in\Sigma$.

(iii) $L_1\in\mathcal{L},\ L_2\in\mathcal{L} \implies L_1^*\in\mathcal{L}, \quad L_1\cdot L_2\in\mathcal{L}, \quad L_1\cup L_2\in\mathcal{L}$.

Satz 2.3 Eine Sprache R über Σ ist regulär genau dann, wenn es einen regulären Ausdruck r mit $R = L(r)$ gibt.

Beweis: Sei $\mathcal{L}(\Sigma) := \{L(r) \mid r \text{ ist ein regulärer Ausdruck über } \Sigma\}$.

$Reg_\Sigma \subseteq \mathcal{L}(\Sigma)$: Beachte, daß $\mathcal{L}(\Sigma)$ abgeschlossen ist unter den Operationen $\emptyset, 1, \{a\}$ für $a\in\Sigma, +, \cdot$ und *. Da Reg_Σ die kleinste Klasse $\mathcal{L}\subseteq 2^{\Sigma^*}$ ist, die unter diesen Operationen abgeschlossen ist, gilt $Reg_\Sigma\subseteq\mathcal{L}(\Sigma)$.

$\mathcal{L}(\Sigma)\subseteq Reg_\Sigma$: Für jeden regulären Ausdruck r über Σ zeigen wir durch Induktion, daß $L(r)\in Reg_\Sigma$.

Induktionsanfang: $L(0) = \emptyset \in Reg_\Sigma$, $L(1) = \{\varepsilon\}\in Reg_\Sigma$ und $L(a) = \{a\}\in Reg_\Sigma$ auf Grund der Definitionen.

Induktionsschritt: Seien schon $L(r)\in Reg_\Sigma$ und $L(s)\in Reg_\Sigma$. Dann gelten:

$$\begin{aligned} L(r\cdot s) &= L(r)\cdot L(s) \ \in Reg_\Sigma, \\ L(r+s) &= L(r)\cup L(s) \ \in Reg_\Sigma, \\ L(r^*) &= L(r)^* \ \in Reg_\Sigma, \end{aligned}$$

da Reg_Σ abgeschlossen ist unter $\cdot, +$ und * auf Grund der Definition. □

Die ALGEBRA DER REGULÄREN SPRACHEN ÜBER Σ,

$$Reg_\Sigma^{\mathcal{L}} := (Reg_\Sigma, \cup, \emptyset, \cdot, \{\varepsilon\}, ^*, \{a\}_{a\in\Sigma}),$$

besteht aus der Menge Reg_Σ und der Einschränkung der angegebenen Operationen von $\mathcal{L}_\Sigma$ auf diese Teilmenge von 2^{Σ^*}. Man nennt $Reg_\Sigma^{\mathcal{L}}$ auch die Algebra der regulären EREIGNISSE, da sie von S.C.Kleene ursprünglich zur Darstellung des Verhaltens von Nervenzellen verwendet wurde.

$Reg_\Sigma^{\mathcal{L}}$ ist eine echte Unteralgebra von $\mathcal{L}_\Sigma$. Es gibt (für $\Sigma \neq \emptyset$) zwar überabzählbar viele Sprachen über Σ, aber nur abzählbar viele reguläre Ausdrücke. Wir können mit regulären Ausdrücken also bei weitem nicht alle Sprachen bezeichnen. (Dasselbe gilt für die später betrachteten kontextfreien Grammatiken und überhaupt alle Sprachdefinitionen mit endlichen Mitteln.)

Beispiel 2.4 (i) $w \in \Sigma^* \Longrightarrow \{w\}$ ist regulär: $L(w) = \{w\}$.

(ii) $L \subseteq \Sigma^*$ endlich $\Longrightarrow L$ ist regulär: $L = \{w_1, \ldots, w_k\} = L(w_1 + \ldots + w_k)$.

(iii) Σ^* ist regulär: $\Sigma^* = L((a_1 + \ldots + a_n)^*)$, für $\Sigma = \{a_1, \ldots, a_n\}$.

(iv) In Beschreibungen von Programmiersprachen werden oft Teilsprachen durch reguläre Ausdrücke angegeben. Dabei benutzt man $r \mid s$ statt $r + s$ und definiert mehrere Sprachen simultan (was noch genauer betrachtet werden muß). Zahlausdrücke und Bezeichner definiert man etwa so:

digit	:=	$0 \vert 1 \vert \ldots \vert 9$
letter	:=	$a \vert \ldots \vert z \vert A \vert \ldots \vert Z$
number	:=	$((\varepsilon \mid -)(1 \vert \ldots \vert 9)$ digit $^* \mid 0)(\varepsilon \mid \times(1 \vert \ldots \vert 9)$ digit $^* \mid \times 0)$
identifier	:=	letter (letter | digit)*

(v) In der Unix-Shell werden Dateinamen mit regulären Ausdrücken beschrieben, sodaß Befehle sehr kurz formuliert werden können. Beispielsweise bedeutet $ls\ a * .asm$ dann: "Liste alle Assembler-Dateien, deren Namen mit a beginnen, auf". Assembler-Dateien haben in der Regel das Suffix ".asm".

Die regulären Sprachen erhält man natürlich auch als den Abschluß *aller endlichen* Sprachen unter Vereinigung, Sprachprodukt und Sprachiteration.

Sei allgemeiner $\mathcal{K} = (K, +, 0, \cdot, 1, ^*)$ eine Menge K mit ausgezeichneten Elementen $0, 1 \in K$, zweistelligen Operationen $+$ und $\cdot$ und einer einstelligen Operation * (mit den in

Abschnitt 3.2 angegebenen Eigenschaften). Die VON $A \subseteq K$ ERZEUGTE REGULÄRE UNTERALGEBRA $Reg_A^{\mathcal{K}}$ von $\mathcal{K}$ sei die Menge

$$Reg_A := \bigcap \{ M \mid A \subseteq M \subseteq K,\ 0,1 \in M,\ M+M \subseteq M,\ M \cdot M \subseteq M,\ M^* \subseteq M \}$$

zusammen mit den Einschränkungen der Operationen aus $\mathcal{K}$ auf Reg_A.

Erlaubt man beim Aufbau der regulären Ausdrücke statt der Konstanten $a \in \Sigma$, die fest interpretiert sind, Variable $x_1, x_2, \ldots$, so gilt analog zu Satz 2.3:

$$Reg_A^{\mathcal{K}} = \{ r^{\mathcal{K}}(a_1,\ldots,a_m) \mid r(x_1,\ldots,x_m) \text{ ein regulärer Ausdruck},\ a_1,\ldots,a_m \in A \}.$$

Dabei ist $r^{\mathcal{K}}(a_1,\ldots,a_m)$ der WERT des Ausrucks r in $\mathcal{K}$, definiert durch

$$\begin{array}{lcllcl}
0^{\mathcal{K}}(a_1,\ldots,a_m) & := & 0, & (r+s)^{\mathcal{K}}(a_1,\ldots,a_m) & := & r^{\mathcal{K}}(a_1,\ldots,a_m) + s^{\mathcal{K}}(a_1,\ldots,a_m), \\
1^{\mathcal{K}}(a_1,\ldots,a_m) & := & 1, & (r\cdot s)^{\mathcal{K}}(a_1,\ldots,a_m) & := & r^{\mathcal{K}}(a_1,\ldots,a_m) \cdot s^{\mathcal{K}}(a_1,\ldots,a_m), \\
x_i^{\mathcal{K}}(a_1,\ldots,a_m) & := & a_i, & (r^*)^{\mathcal{K}}(a_1,\ldots,a_m) & := & (r^{\mathcal{K}}(a_1,\ldots,a_m))^*.
\end{array}$$

Die Algebra $Reg_\Sigma^{\mathcal{L}}$ der regulären Sprachen über Σ ist gerade die von Σ erzeugte reguläre Unteralgebra von $\mathcal{L} = \mathcal{L}_\Sigma$.

2.2 Reguläre Ausdrücke als zweistellige Relationen

Sei M eine Menge $\neq \emptyset$ und $2^{M\times M}$ die Menge aller zweistelligen Relationen auf M. Darauf hatten wir die speziellen Relationen $\emptyset$ und $1_M = Id_M$ definiert, sowie die Operationen

$$\begin{array}{lcl}
R_1 \cup R_2 & := & \{ (a,b) \in M\times M \mid (a,b) \in R_1 \text{ oder } (a,b) \in R_2 \}, \\
R_1 \circ R_2 & := & \{ (a,c) \in M\times M \mid \text{Es gibt } b \in M \text{ mit } (a,b) \in R_1 \text{ und } (b,c) \in R_2 \}, \\
R^* & := & \bigcap \{ S \subseteq M\times M \mid R \subseteq S,\ S \text{ ist reflexiv und transitiv } \}.
\end{array}$$

Wie für L^* bei Sprachen sieht man ähnlich wie im Beweis von Lemma 1.2:

$$R^* = \bigcup_{n=0}^{\infty} R^n \qquad \text{mit } R^0 := 1_M,\ R^{n+1} := R^n \circ R.$$

Sei für jedes $a \in \Sigma$ eine Relation $R_a \subseteq M \times M$ gegeben, z.B. $\{(a,a)\}$ im Falle $\Sigma \subseteq M$. Dann können wir jeden regulären Ausdruck r über Σ in der Algebra

$$\mathcal{R}_M := (2^{M\times M}, \emptyset, \cup, 1_M, \circ, {}^*, R_{a_1}, \ldots, R_{a_n})$$

interpretieren durch

$$\begin{array}{lcllcl}
R(0) & := & \emptyset & R(r\cdot s) & := & R(r) \circ R(s) \\
R(1) & := & 1_M & R(r+s) & := & R(r) \cup R(s) \\
R(a) & := & R_a \quad \text{für } a \in \Sigma, & R(r^*) & := & R(r)^*.
\end{array}$$

Man beachte, daß für diese Interpretation Gleichungen gelten können, die bei der Interpretation durch Sprachen nicht gelten: Für verschiedene $a, b, c \in \Sigma$ kann etwa $R(a \cdot b) = R(c)$ sein, während stets $L(a \cdot b) = \{ab\} \neq \{c\} = L(c)$ ist.

Die von $A := \{ R_a \mid a \in \Sigma \}$ erzeugte reguläre Unteralgebra

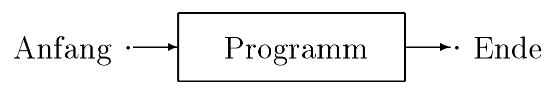
$$Reg_A^{\mathcal{R}} = \{ r^{\mathcal{R}}(R_{a_1}, \dots, R_{a_m}) \mid r(x_1, \dots, x_m) \text{ ein regulärer Ausdruck}, \ R_{a_1}, \dots, R_{a_m} \in A \}$$

von $\mathcal{R}_M$ ist die Algebra der von A erzeugten REGULÄREN RELATIONEN AUF M. Oft schreiben wir auch $\mathcal{R}_M$ für $(2^{M \times M}, \cup, \emptyset, \circ, 1_M, {}^*)$ ohne ausgezeichnete Relationen.

2.3 Reguläre Ausdrücke als Programme

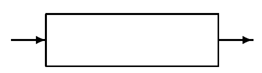
Wir können reguläre Ausdrücke auch als eine abstrakte Form von (sequentiellen, nicht deterministischen) Programmen ansehen, oder als Aktionen, die man ausführen soll.

Wir geben Programme als Flußdiagramme an. Dies sind hierarchisch aufgebaute Graphen mit einem ausgezeichnteten Anfangs- und Endpunkt:

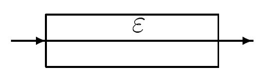


Die Grundprogramme oder BASISAKTIONEN $P(a)$, für $a \in \Sigma$, werden nicht weiter analysiert; ihre Namen a werden zur Markierung der Kanten des Graphen verwendet.

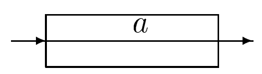
$P(0) =$ loop forever $=$



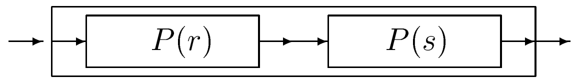
$P(1) =$ skip $=$



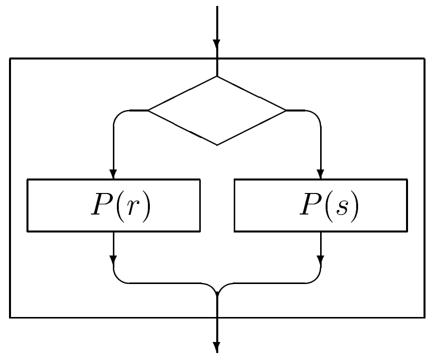
$P(a) =$ do a $=$



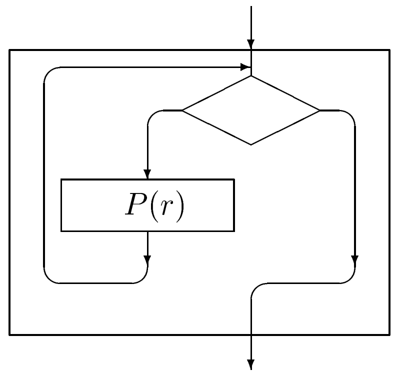
$P(r \cdot s) =$ (first do r and then do s) $=$



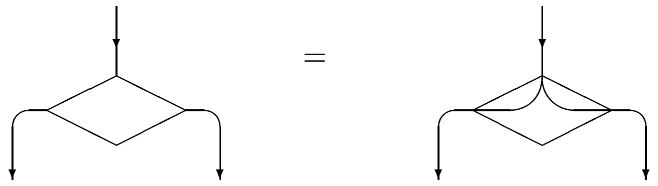
$P(r + s) =$ (do r or do s) $=$



$P(r^*) =$ (do r repeatedly) $=$ (choose $n \in I\!N$ and then do r n times) $=$



Da wir bei der Parallelschaltung $P(r + s)$ und bei der Iteration $P(r^*)$ die Wahl der Ausführung nicht von einer Fallunterscheidung abhängig machen, sind die Verzweigungsrauten wie in



zu verstehen. Wie man hier Tests einfügt, um von nicht-deterministischen zu deterministischen Programmen zu kommen, tragen wir im folgenden Abschnitt nach.

2.4 Der Zusammenhang zwischen den Interpretationen

Zwischen den drei bisher betrachteten Interpretationen regulärer Ausdrücke besteht ein wichtiger Zusammenhang.

Wir können erstens die regulären Ausdrücke r über Σ mit den Flußdiagrammen $P(r)$ oder den entsprechenden Programmen identifizieren. Dabei betrachten wir alle Flußdiagramme als gleich, die sich nur durch Vergrößerungen und Streckungen voneinander unterscheiden. Davon abgesehen ist der Aufbau der Flußdiagramme der gleiche wie der der regulären Ausdrücke. Wir sprechen daher statt von regulären Ausdrücken von den REGULÄREN PROGRAMMEN über den Grundbefehlen Σ.

Betrachtet man die regulären Ausdrücke als Programme, so entsprechen die Sprach- und die Relationeninterpretation zwei verschiedenen Arten, die Bedeutungen von Programmen zu erklären:

Die OPERATIONALE BEDEUTUNG oder das ENDLICHE VERHALTEN (englisch: TRACE) eines Programms ist die Gesamtheit aller Folgen atomarer Befehle, die in terminierenden Ausführungen des Programms durchlaufen werden.[1] Diese operationale Bedeutung entspricht sehr direkt dem Verhalten einer Maschine, die das Programm ausführt. Wir setzen also:

Operationale Bedeutung von r
= *Menge aller Folgen von Grundbefehlen, die einen Weg durch das Flußdiagramm $P(r)$ zu r vom Start- zum Endpunkt markieren,*
= $\mathcal{L}(r)$.

Die DENOTATIONALE BEDEUTUNG eines Programms ist die Beziehung zwischen Ein- und Ausgaben des Programms. Von Ein- und Ausgaberoutinen abgesehen, ist eine EINGABE einfach eine Belegung des Speichers *vor* und eine AUSGABE eine Belegung des Speichers *nach* der Programmausführung.

Eine Belegung des Speichers, oder ein SPEICHERZUSTAND, besteht im konkreten Fall in der Angabe, welche Werte in welchen Speicherzellen stehen; für den Programmierer genügt es zu wissen, welche Werte die im Programm vorkommenden Variablen haben. Da wir hier die Grundbefehle (z.B. Zuweisungen $x := y + 3$) der Programme nicht weiter analysieren, betrachten wir eine beliebige Menge M von Speicherzuständen.

Denotationale Bedeutung von r
= $\{(m, n) \in M \times M \mid$ *der Speicherzustand m wird durch eine Ausführung von r in den Speicherzustand n überführt* $\}$
= $\mathcal{R}(r)$.

[1] Beachte, daß wegen der nichtdeterministischen Fallunterscheidung in $(r+s)$ und der nicht festgelegten Anzahl von Iterationen bei r^* verschiedene Abläufe desselben Programms möglich sind.

Der folgende einfache Satz beschreibt einen Zusammenhang zwischen der operationalen und der denotationalen Semantik regulärer Programme: die denotationale Bedeutung eines Programms setzt sich aus den Speicherveränderungen der Grundoperationen entsprechend den möglichen Programmabläufen zusammen.

Satz 2.5 Zu jedem $a \in \Sigma$ sei eine Relation $\vdash_a \subseteq M \times M$ gegeben. Definiere auf der Algebra $\mathcal{R} = (2^{M\times M}, \cup, 0_M, \circ, 1_M, {}^*, \vdash_a)_{a\in\Sigma}$ zu jedem $L \subseteq \Sigma^*$ eine Relation $\vdash_L \subseteq M \times M$ durch

$$\vdash_L := \bigcup\{\vdash_w \mid w \in L\}, \qquad \text{wobei } \vdash_\varepsilon := 1_M, \qquad \vdash_{va} := \vdash_v \circ \vdash_a .$$

Dann gilt $\mathcal{R}(r) = \vdash_{L(r)}$ für jeden regulären Ausdruck r über Σ.

Beweis: Man sieht direkt, daß $\mathcal{R}(0) = \vdash_{L(0)}$, $\mathcal{R}(1) = \vdash_{L(1)}$ und $\mathcal{R}(a) = \vdash_{L(a)}$. Für zusammengesetzte Ausdrücke folgt die Behauptung aus folgenden leicht nachzurechnenden Eigenschaften:

$$\begin{array}{lcl} \vdash_{(L_1\cup L_2)} & = & \vdash_{L_1} \cup \vdash_{L_2} \\ \vdash_{(L_1\cdot L_2)} & = & \vdash_{L_1} \circ \vdash_{L_2} \\ \vdash_{L^*} & = & (\vdash_L)^* \end{array}$$

$\square$

Da die Wahl von M und den Ausgangsrelationen $\vdash_a$ beliebig war, zeigt der Satz, daß die Sprachinterpretation feiner ist als die Relationeninterpretation: *jede* Relationeninterpretation $\mathcal{R}$ läßt sich aus der Sprachinterpretation $\mathcal{L}$ und den Grundrelationen $\mathcal{R}(a)$ durch $\mathcal{R}(r) = \vdash_{L(r)}$ gewinnen.

Folgerung 2.6 Aus $\mathcal{L}(r) = \mathcal{L}(s)$ folgt $\mathcal{R}(r) = \mathcal{R}(s)$.

Die Umkehrung gilt natürlich nicht: zwei Programme können dieselbe Relation oder Funktion berechnen, ohne daß ihre möglichen Programmabläufe übereinstimmen.

Auch einzelne Programmabläufe $v, w \in \Sigma^*$ können dieselbe Ein/-Ausgabebeziehung bestimmen, ohne daß die Abläufe gleich sind: $\vdash_v = \vdash_w \not\Rightarrow v = w$.

Interpretiert man reguläre Ausdrücke als Programme, so schreibt man oft $(r; s)$ statt $(r \cdot s)$ für die sequentielle Verkettung von Programmen.

Bemerkung 2.7 Bei den Programmen wurde bisher vom Inhalt der Tests abstrahiert. Um Programme mit Tests und deterministische Programme zu erfassen, muß man die regulären Ausdrücke erweitern. Seien dazu neben den Basisaktionen AKTION $= \Sigma$ eine endliche Menge TEST $= \{b_1, \overline{b}_1, \ldots, b_k, \overline{b}_k\}$ von Basistests gegeben. Verallgemeinerte reguläre Programme kann man dann durch folgende Definitionen einführen:[2]

$$\begin{array}{ll} \mathit{test} := & b_1 \mid \ldots \mid b_k \mid \overline{b}_1 \mid \ldots \mid \overline{b}_k \\ & \mid (\mathit{test} \vee \mathit{test}) \mid (\mathit{test} \wedge \mathit{test}) \\ \mathit{program} := & a_1 \mid \ldots \mid a_n \mid \mathit{loop} \mid \mathit{skip} \\ & \mid (\mathit{test} \cdot \mathit{program}) \\ & \mid (\mathit{program} + \mathit{program}) \\ & \mid (\mathit{program} \cdot \mathit{program}) \\ & \mid \mathit{program}^* \end{array}$$

Dabei ist die neue Konstruktionsmöglichkeit als

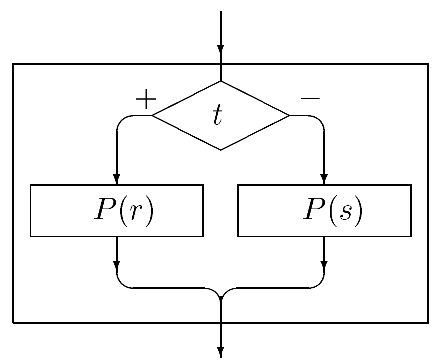
$$(\mathit{test} \cdot \mathit{program}) := (\text{if } \mathit{test} \text{ then do } \mathit{program})$$

zu verstehen. Für so zusammengesetzte t definiert man den komplementären Test $\overline{t}$ durch

$$\overline{(t_1 \vee t_2)} := (\overline{t}_1 \wedge \overline{t}_2) \qquad \text{und} \qquad \overline{(t_1 \wedge t_2)} := (\overline{t}_1 \vee \overline{t}_2).$$

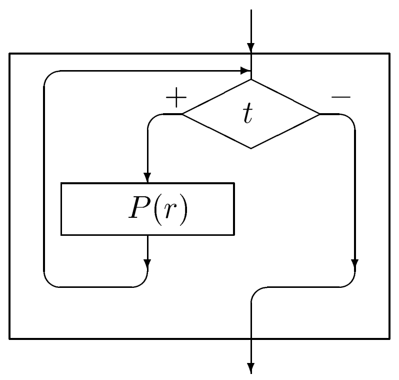
DETERMINISTISCHE (REGULÄRE) PROGRAMME sind dann diejenigen, bei denen die regulären Operationen $+$ und * nur auf folgende Weisen verwendet werden:

$P(t \cdot r + \overline{t} \cdot s) = (\text{if } t \text{ then do } r \text{ else do } s) =$

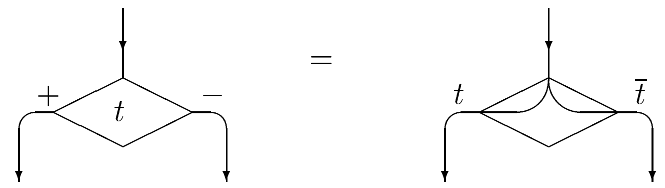


[2]Diese Kurzschreibweise soll heißen: *test* ist die kleinste Menge T mit (i) $\{b_1, \ldots, b_k, \overline{b}_1, \ldots, \overline{b}_k\} \subseteq T$, (ii) Für $t \in T$ und $s \in T$ ist auch $(t \vee s) \in T$ und $(t \wedge s) \in T$. Entsprechend für die Programme.

$P((t \cdot r)^* \cdot \overline{t}) = (\text{while } t \text{ do } r) =$



Hierbei sind die Tests so zu verstehen:



Um die angegebene denotationelle Semantik auf Programme mit Tests auszudehnen, können wir wie folgt vorgehen. Sei $\Delta = \{b_1, \ldots, b_k, \overline{b_1}, \ldots, \overline{b_k}\}$ eine zu Σ disjunkte Menge von atomaren Tests.

In der Relationeninterpretation ordnen wir jedem Test t die Menge $Test(t) \subseteq M$ derjenigen Speicherzustände zu, die den Test t erfüllen. Das ist gleichwertig damit, daß den Tests die folgende Relation zugeordnet wird:

$$\vdash_t \; := \; 1_{Test(t)} \; = \; \{(m,m) \mid m \in Test(t)\} \; \subseteq \; M \times M.$$

Basistests b und ihre Komplementärtests $\overline{b}$ sind dual zueinander zu interpretieren:

$$\vdash_{\overline{b}} \; := \; 1_M - \vdash_b \; = \{(m,m) \mid m \notin Test(b)\}.$$

Für die Sprachinterpretation ist zu beachten, daß es – im Unterschied zur Relationeninterpretation – keine Sprachen L mit $\emptyset = L(0) \subset L \subset L(1) = \{\varepsilon\}$ gibt, die als $L(t)$ in Frage kämen. Man kann die Programme aber als formale Sprachen $L \subseteq (\Sigma \cup \Delta)^*$ interpretieren; dabei merkt man sich im Programmablauf $w \in L(r) \subseteq (\Sigma \cup \Delta)^*$ auch, welche Basistests durchlaufen wurden.

Aufgaben

Aufgabe 2.1 Zeige für beliebige Sprachen $L, L_1, L_2 \subseteq \Sigma^*$:

(a) Wenn $L_1 \subseteq L_2$ ist, so ist auch $L_1^* \subseteq L_2^*$.

(b) $L^* = \bigcup_{n=0}^{\infty} L^n$, wobei L^n durch $L^0 := \{\epsilon\}$ und $L^{n+1} := L^n \cdot L$ definiert ist.

(c) $L \cdot L^* = L^* \cdot L$.

(d) $(L^*)^* = L^*$.

Zeige, daß diese Gleichungen auch für Relationen $L \subseteq M \times M$ gelten.

Aufgabe 2.2 Zeige durch Nachrechnen, daß folgende Gleichungen zwischen Sprachen wahr sind, für beliebige Sprachen $L_1, L_2, L_3 \subseteq \Sigma^*$.

(a) $(L_1 \cup L_2)^* = (L_1^* \cdot L_2^*)^*$.

(b) $L_1(L_2 \cup L_3) = (L_1 \cdot L_2) \cup (L_1 \cdot L_3)$.

(c) $\{\varepsilon\} \cup (L_1^* \cdot L_1^*) \cup L_1 = L_1^*$.

Zeige, daß folgende Gleichungen bei der Sprachinterpretation nicht für alle regulären Ausdrücke r, s über $\Sigma = \{a, b\}$ gelten:

(a) $(r + s)^* = r^* + s^*$.

(b) $((r \cdot s) + r)^* \cdot r = r \cdot ((r \cdot s) + r)^*$.

Für *spezielle* Ausdrücke r und s können diese Gleichungen aber doch gelten. Gib auch dafür Beispiele an.

Aufgabe 2.3 Zeige, daß die Klasse der regulären Sprachen unter Substitution abgeschlossen ist. Das heißt:

Ist R und für jedes $a \in \Sigma$ auch L_a eine reguläre Sprache über Σ, so ist auch

$$L_R := \bigcup \{\, L_w \mid w \in R \,\} \qquad \text{mit} \quad L_{va} := L_v \cdot L_a \text{ und } L_\varepsilon := \{\varepsilon\}$$

eine reguläre Sprache über Σ.

Hinweis: Benutze reguläre Ausdrücke r für R und r_a für L_a.

Aufgabe 2.4 Gib einen regulären Ausdruck an, der die Sprache

$$\{w \in \{a,b,c\}^* \mid ac \text{ ist kein Teilwort von } w\}$$

beschreibt.

Aufgabe 2.5 Sei $M = I\!N^3$ die Menge aller Tripel (k,m,n) von natürlichen Zahlen. Man stelle sich vor, daß diese Zahlen im Datenspeicher einer Maschine unter den Adressen x, y, und z gespeichert sind. Ist $f : I\!N^3 \to I\!N$, so hat die Zuweisung

$$\cdot \to \boxed{x \leftarrow f(x,y,z)} \to \cdot$$

als relationale Bedeutung die folgende Relation zwischen dem Speicherzustand vor und nach Ausführung der Zuweisung:

$$\boxed{x \leftarrow f(x,y,z)}^{R(M)} := \{((l,m,n),(f(l,m,n),m,n)) \mid l,m,n \in I\!N\}.$$

Betrachte solche Anweisungen $(x \leftarrow f(x,y,z))$ als Elemente eines Alphabets Σ von atomaren Programmen.

(a) Seien $a := (y \leftarrow x+z)$ und $b := (x \leftarrow x \dot{-} 1)$ atomare Programme. Die Bedeutung der Differenz sei für $n > k$ durch $k \dot{-} n := 0$ festgelegt. Berechne die relationale Bedeutung $(a;b)^{R(M)}$ des zusammengesetzten Programms

$$\cdot \to \boxed{\boxed{y \leftarrow x+z} \to \cdot \to \boxed{x \leftarrow x \dot{-} 1}} \to \cdot$$

(b) Nun berechne die relationale Bedeutung des Programms

$$(\textit{while } x > 0 \textit{ do } ((y \leftarrow x+z);(x \leftarrow x \dot{-} 1)))$$

in zwei Schritten.

Berechne zuerst die relationale Bedeutung $(r^*)^{R(M)}$ des regulären Ausdrucks

$$r^* := ((y \leftarrow x+z);(x \leftarrow x \dot{-} 1))^*.$$

Die Ausführung eines Tests wie $(x > 0?)$ ändert den gegenwärtigen Zustand nicht, sondern verhindert weitere Zustandsübergänge, falls die Bedingung nicht erfüllt ist. Das wird durch Teilrelationen der Identität auf M wie folgt modelliert. Für den Test $(x > 0?)$ auf die Eigenschaft

$$B := \{(k,m,n) \mid k,m,n \in I\!N, k > 0\}$$

(von Zuständen!) benutze die Relationen

$$\begin{aligned} 1_B &:= \{((k,m,n),(k,m,n)) \mid k,m,n \in I\!N, k > 0\}, \\ 1_{\overline{B}} &:= \{((k,m,n),(k,m,n)) \mid k,m,n \in I\!N, k = 0\} \end{aligned}$$

und interpretiere dann das angegebene *while*-Programm über

$$(\textit{while } B \textit{ do } r)^{R(M)} := ((1_B;r)^*;1_{\overline{B}})^{R(M)}.$$

Aufgabe 2.6 Definiere zu jedem regulären Ausdruck r über Σ und jedem Buchstaben $a \in \Sigma$ rekursiv über den Aufbau von r

(a) einen regulären Ausdruck $(r - 1)$ mit $L((r - 1)) = L(r) - \{\varepsilon\}$,

(b) einen regulären Ausdruck $(r - a)$ mit $L((r - a)) = L(r) - \{a\}$.

3 Die Gleichungstheorie regulärer Algebren

Von den sogenannten Standard-Interpretationen kann man abstrahieren: statt

$$\mathcal{L}_\Sigma = (2^{\Sigma^*}, \cup, \emptyset, \cdot, \{\varepsilon\}, \{a\}, {}^*)_{a\in\Sigma}$$

oder der Relationenalgebra

$$R_M = (2^{M\times M}, \cup, \emptyset, \circ, 1_M, R_a, {}^*)_{a\in\Sigma}$$

mit den spezifischen Operationen wie Verkettung bzw. Relationenprodukt kann man beliebige Algebren mit gewissen Eigenschaften betrachten; dabei wird auf die Angabe ausgezeichneter Elemente $a_1, \ldots, a_n$ verzichtet.

Es gibt unter dem Namen REGULÄRE ALGEBRA mehrere verschiedene Definitionen für solche Algebren, da es nicht einfach ist, die wesentlichen Eigenschaften der Iteration * anzugeben.

Man ist hauptsächlich daran interessiert, welche regulären Ausdrücke dieselbe Sprache definieren. Dazu nennt man

$$REG_\Sigma \ := \ \{(r,s) \mid \mathcal{L}_\Sigma(r) = \mathcal{L}_\Sigma(s), r,s \text{ reguläre Ausdrücke über } \Sigma\}$$

die GLEICHUNGSTHEORIE REGULÄRER ALGEBREN.

3.1 Idempotente Halbringe: die Axiomatisierung von $+,\ 0,\ \cdot,\ 1$

Unproblematisch sind die Eigenschaften der Addition und Multiplikation, sowie ihrer neutralen Elemente:

Definition 3.1 Ein IDEMPOTENTER HALBRING ist eine Algebra $\mathcal{A} = (A, +^A, 0^A, \cdot^A, 1^A)$, deren Operationen $+^A$, 0^A, $\cdot^A$ und 1^A die folgenden Anforderungen erfüllen, für alle Elemente a, b und c aus A:

(i) $+$ ist assoziativ, mit beidseitig neutralem Element 0 :

$$(a+b)+c = a+(b+c), \qquad 0+a = a+0 = a$$

(ii) $+$ ist kommutativ und idempotent:

$$a+b = b+a, \qquad a+a = a$$

(iii) $\cdot$ ist assoziativ, mit beidseitig neutralem Element 1 :

$$(a \cdot b) \cdot c = a \cdot (b \cdot c), \qquad a \cdot 1 = 1 \cdot a = a$$

(iv) $\cdot$ ist beidseitig distributiv bezüglich + :

$$a \cdot (b + c) = (a \cdot b) + (a \cdot c), \qquad (b + c) \cdot a = (b \cdot a) + (c \cdot a)$$

(v) 0 ist beidseitig vernichtendes Element bezüglich $\cdot$:

$$0 \cdot a = a \cdot 0 = 0$$

Diese Axiome bezeichnen wir auch als die THEORIE DER IDEMPOTENTEN HALBRINGE.

Proposition 3.2 Ist $\mathcal{A} = (A, +^A, 0^A, \cdot^A, 1^A)$ ein idempotenter Halbring, so gelten:

(i) Durch $a \leq^A b :\Longleftrightarrow a +^A b = b$ wird $(A, \leq^A)$ eine partielle Ordnung.

(ii) Die Operationen + und $\cdot$ sind MONOTON BEZÜGLICH $\leq$, d.h. für alle $a, b \in A$ gilt:

$$a \leq b \quad \Longrightarrow \quad x + a \leq x + b, \quad x \cdot a \leq x \cdot b, \quad \text{und} \quad a \cdot x \leq b \cdot x.$$

Beweis: (ii) Sei $a \leq b$, also $a + b = b$. Dann ist für $x \in A$

$$(x + a) + (x + b) = (x + x) + (a + b) = (x + x) + b = x + b,$$

also $x + a \leq x + b$. Der Beweis der übrigen Aussagen bleibt dem Leser überlassen. $\square$

Schwierig ist es, die Iteration durch einfache Eigenschaften vollständig zu charakterisieren. Wir geben drei Axiomatisierungen an.

3.2 Die Axiomatisierung der Iteration * nach Salomaa

Eine wesentliche Eigenschaft der Iteration *, die A.Salomaa[23] als Axiom annimmt, ist durch die eindeutige Lösbarkeit gewisser Gleichungen bei der Sprachinterpretation motiviert:

Lemma 3.3 (Arden) Zu $A, B \subseteq \Sigma^*$ mit $\varepsilon \notin A$ ist A^*B bzw. BA^* die einzige Menge $X \subseteq \Sigma^*$ mit

$$X = A \cdot X \cup B \qquad \text{bzw.} \quad X = X \cdot A \cup B. \tag{1}$$

Beweis: A^*B ist eine Lösung von $X = AX \cup B$, da

$$A^*B = (\{\varepsilon\} \cup AA^*)B = B \cup AA^*B = A(A^*B) \cup B.$$

Sei $X \subseteq \Sigma^*$ eine Lösung von $X = AX \cup B$. Für $A^*B \subseteq X$ genügt es zu zeigen, daß $A^nB \subseteq X$ für alle $n \in \mathbb{N}$ gilt. Für $n = 0$ ist $A^0B = B \subseteq AX \cup B \subseteq X$. Ist für ein $n \in \mathbb{N}$ schon $A^nB \subseteq X$, so ist auch

$$A^{n+1}B = A(A^nB) \subseteq AX \subseteq AX \cup B \subseteq X.$$

Nach Induktion ist daher $A^nB \subseteq X$ für jedes $n \in \mathbb{N}$.

Für die Umkehrung $X \subseteq A^*B$ sei $w \in X \subseteq AX \cup B$. Falls es kein $v \in X$ mit $|v| < |w|$ gibt, so ist $w \notin AX$ wegen $\varepsilon \notin A$, also $w \in B \subseteq A^*B$. Im anderen Fall gilt nach Induktion über $|w|$ für alle $v \in X$ mit $|v| < |w|$ schon $v \in A^*B$. Für $w \in B$ ist $w \in A^*B$ klar. Für $w \in AX$ gibt es $u \in A$ und $v \in X$ mit $w = uv$. Wegen $\varepsilon \notin A$ muß $|v| < |w|$ sein, also $v \in A^*B$ und daher auch $w = uv \in A(A^*B) \subseteq A^*B$. $\square$

Der Beweis zeigt, daß ohne die Voraussetzung $\varepsilon \notin A$ noch gilt:

$$A(A^*B) \cup B = A^*B \quad \wedge \quad \forall X(AX \cup B \subseteq X \longrightarrow A^*B \subseteq X).$$

Die Axiomatisierung von Salomaa besteht nun aus folgenden Forderungen an Algebren $\mathcal{A} = (A, +^A, 0^A, \cdot^A, 1^A, {}^{*A})$:

(S 0) Durch Weglassen von * wird $\mathcal{A}$ ein idempotenter Halbring.

(S 1) Für alle $a \in A$ gilt:

(i) $0^* = 1$

(ii) $a^* = 1 + (a^* \cdot a) = 1 + (a \cdot a^*)$

(iii) $a^* = (1 + a)^*$

(S 2) Für alle $a \in A$ mit $1 \not\leq a$ und alle $b \in A$ haben die Gleichungen $x = ax + b$ und $x = xa + b$ in $\mathcal{A}$ jeweils höchstens eine Lösung, d.h. für alle $a, b, c \in A$ gilt:

(iv) Falls $1 \not\leq a$ und $c = (c \cdot a) + b$ ist, so ist $c = b \cdot a^*$.

(v) Falls $1 \not\leq a$ und $c = (a \cdot c) + b$ ist, so ist $c = a^* \cdot b$.

Aus den anderen Axiomen folgt schon, daß a^*b eine Lösung von $x = ax + b$ ist:

$$a^*b = (aa^* + 1)b = (aa^*)b + b = a(a^*b) + b.$$

Entsprechend folgt aus den anderen Axiomen, daß ba^* eine Lösung von $x = xa + b$ ist.

Man kann diese Axiomatisierung als einen KALKÜL zum Rechnen mit regulären Ausdrücken über Σ darstellen. Dazu erlaubt man für a, b, c reguläre Ausdrücke r, s, t. Aus den Axiomen für idempotente Halbringe und denen in (S 1) entstehen dann Gleichungen zwischen regulären Ausdrücken, die als bewiesen gelten (als Beweise der Länge 1). Die Implikationen in (S 2) werden zu den Schlußregeln[3]

$$\text{(Gen)} \quad \frac{r = (r \cdot s) + t}{r = t \cdot s^*}, \qquad \frac{r = (s \cdot r) + t}{r = s^* \cdot t}, \qquad \text{falls } \varepsilon \notin L(s),$$

die auch als "(Generalisierungs-) Regeln von Arden" bekannt sind. Dazu kommen noch die üblichen Gleichheitsaxiome und Regeln zur Substitution von Gleichem für Gleiches:

$$\text{(Subst)}_1 \quad \frac{r = s \qquad t = u}{t[s/r] = u}, \qquad \text{(Subst)}_2 \quad \frac{r = s \qquad t = u}{t = u[s/r]}.$$

Mit $t[s/r]$ ist der Ausdruck gemeint, der aus t entsteht, wenn man darin alle Vorkommen von r durch s ersetzt.

Nach Salomaa ist dieser Kalkül nicht nur KORREKT, sondern auch VOLLSTÄNDIG für die Gleichheit bei der Sprachinterpretation:

Satz 3.4 (i) (Korrektheit) Falls $r = s$ aus den Gleichungen mit Hilfe der Regeln bewiesen werden kann, so gilt $L(r) = L(s)$.

(ii) (Vollständigkeit) Falls $L(r) = L(s)$ gilt, so kann die Gleichung $r = s$ aus den Axiomen mit Hilfe der Regeln bewiesen werden.

Beweis: i) Man rechnet leicht nach, daß die Sprachinterpretation die Eigenschaften eines idempotenten Halbrings und die Zusatzbedingungen (S 1) für 0, 1 und * erfüllt, z.B.

$$\begin{aligned} L(r^*) = L(r)^* &= \bigcup_{n=0}^{\infty} L(r)^n = L(r)^0 \cup \bigcup_{n=0}^{\infty} L(r) \cdot L(r)^n \\ &= L(1) \cup L(r) \cdot L(r^*) = L(1 + rr^*). \end{aligned}$$

Die Korrektheit der Regeln (Gen) folgt natürlich aus Lemma 3.3. Ist $\varepsilon \notin L(s)$ und die Prämisse etwa der linken Regel erfüllt, d.h. $L(r) = L(t + rs) = L(r)L(s) \cup L(t)$, so ist $L(r)$ die einzige Lösung von $X = X \cdot L(s) \cup L(t)$, also $L(r) = L(t) \cdot L(s)^* = L(t \cdot s^*)$.

Reflexivität, Transitivität und Symmetrie der Gleichheit sind klar. Die Korrektheit der Substitutionsregeln kann man durch Induktion über den Aufbau der regulären Ausdrücke zeigen.

ii) Der Beweis ist schwieriger und muß hier weggelassen werden. □

[3]Die Schreibweise der Regeln bedeutet: jeder Beweis der Gleichungen über dem Querstrich wird durch die Gleichung unterhalb des Querstrichs zu einem Beweis dieser Gleichung fortgesetzt.

Um die Regel (Gen) anwenden zu können, muß man jeweils überprüfen, ob die Nebenbedingung $\varepsilon \notin L(s)$ erfüllt ist. Das kann man wie folgt tun:

Beh.: Für jeden regulären Ausdruck s kann man effektiv feststellen, ob $\varepsilon \in L(s)$.

Bew.: durch Induktion über den Aufbau der regulären Ausdrücke: Vier Fälle sind klar:

$$\varepsilon \notin L(0) = \emptyset, \quad \varepsilon \in L(1) = \{\varepsilon\}, \quad \varepsilon \notin L(a) = \{a\}, \quad \text{und} \quad \varepsilon \in L(r^*).$$

Bei den übrigen beiden Fällen führen wir die Frage auf dasselbe Problem für Teilausdrücke zurück:

$$\begin{array}{lcl} \varepsilon \in L(r+s) & \iff & \varepsilon \in L(r) \text{ oder } \varepsilon \in L(s), \\ \varepsilon \in L(r \cdot s) & \iff & \varepsilon \in L(r) \text{ und } \varepsilon \in L(s). \end{array}$$

Nach Induktion kann man feststellen, ob hierbei die rechten Seiten gelten. □

Die Regel (Gen) hat den Nachteil, daß man für s nicht jeden regulären Ausdruck einsetzen kann, da man die Nebenbedingung beachten muß[4]. Man würde diese Regel daher gern durch eine Reihe von Gleichungen für die Operation * ersetzen. Im Unterschied zu den anderen Operationen ist das aber nicht möglich:

Satz 3.5 (i) (V. N. Redko, 1964) Durch *kein* endliches System von Gleichungen kann die Menge der bei der Sprachinterpretation gültigen Gleichungen vollständig axiomatisiert werden.

(ii) (H. Conway, 1971) Es gibt Interpretationen $\mathcal{M} = (M, +^M, 0^M, \cdot^M, 1^M, {}^{*^M})$, bei denen alle Gleichungen $r = s$ gelten, für die $L(r) = L(s)$ ist, und wo r^* *nicht* die transitive und reflexive Hülle von r in M ist.

Beweis: Wir zeigen nur den Teil ii): Die REFLEXIVE, TRANSITIVE HÜLLE VON r IN M ist das $\leq$-kleinste $s \in M$ mit $1 \leq s$ (d.h. s ist reflexiv), $ss \leq s$ (d.h. s ist transitiv) und $r \leq s$ (d.h. s umfaßt r), wobei $r \leq s$ durch $r +^M s = s$ definiert ist.

Betrachte folgende Interpretation $\mathcal{M}$, die sogenannte CONWAY'SCHE UNSTETIGKEIT: $M = \{0, 1, 2, 3\}$, mit $0^M := 0$, $1^M := 1$, $0^* := 1$, $1^* := 1$, $2^* := 3$, $3^* := 3$, sowie

$$m +^M n \quad := \quad \max(m, n),$$

$$m \cdot^M n \quad := \quad \begin{cases} 0, & \text{falls m=0 oder n=0} \\ \max(m, n), & \text{sonst.} \end{cases}$$

Man prüft leicht, daß in $\mathcal{M}$ die Axiome von Salomaa erfüllt sind. Die Regeln (Gen) bzw. die Implikationsaxiome (S 2) fallen wegen der Kommutativität von $\cdot^M$ zusammen zu

$$1 \leq a \ \vee\ \forall x(x = b + x \cdot a \ \rightarrow\ x = b \cdot a^*).$$

Diese Aussage gilt in $\mathcal{M}$ für $a = 1, 2, 3$ wegen $1 \leq^M a$, und für $a = 0$ wegen $b + x \cdot 0 = b = b \cdot 1 = b \cdot 0^*$. Nun ist wegen $1 \leq^M 2$ und $2 \cdot^M 2 = 2 \leq^M 2$ das Element 2 reflexiv und transitiv, also die reflexive und transitive Hülle von 2 in $\mathcal{M}$, aber verschieden von $2^* = 3$! Beachte auch, daß $(\sum_{i=0}^{\infty} 2^i) = 2 <^M 2^*$. □

[4] Anders als die Gleichungen ist die Bedingung $1 \not\leq s$ nicht unter Substitution von regulären Sprachen für Buchstaben abgeschlossen.

3.3 Kleene'sche Algebra: die Axiomatisierung der Iteration * nach Kozen

Die Generalisierungsregel von Salomaa (bzw. das entsprechende Axiom) ist sehr speziell auf die Sprachinterpretation zugeschnitten und gilt nicht für die Relationeninterpretation. Erst 1991 wurde von D. Kozen[12] eine endliche Axiomatisierung angegeben, die auch korrekt für die Relationeninterpretation ist.

Definition 3.6 Eine Algebra $\mathcal{A} = (A, +, 0, \cdot, 1, ^*)$ ist eine Kleene'sche Algebra, wenn

(i) $(A, +, 0, \cdot, 1)$ ein idempotenter Halbring ist, und

(ii) für alle $a, b \in A$ folgende Eigenschaften gelten – mit $a \leq b :\Longleftrightarrow (a + b) = b$ –:

$$\begin{array}{lll} (\textit{KA-1}) & 1 + a \cdot a^* \leq a^* & \wedge \quad \forall x(b + a \cdot x \leq x \longrightarrow a^* \cdot b \leq x) \\ (\textit{KA-2}) & 1 + a^* \cdot a \leq a^* & \wedge \quad \forall x(b + x \cdot a \leq x \longrightarrow b \cdot a^* \leq x) \end{array}$$

Die Erweiterung der Theorie der idempotenten Halbringe um *KA-1* und *KA-2* nennen wir auch die Theorie der Kleene'schen Algebren, kurz *KA*.

Nach *KA-1* ist a^*b das kleinste Element x von K, für das $b + ax \leq x$ gilt. Entsprechend besagt *KA-2*, daß ba^* das kleinste Element x mit $b + xa \leq x$ ist. Beachte, daß diese Axiome nicht nur Gleichungen enthalten, sondern auch Implikationen zwischen Gleichungen. Im Unterschied zum Lemma von Arden hat man:

Lemma 3.7 Ist $\mathcal{K}$ eine Kleene'sche Algebra und sind $a, b \in K$, so ist a^*b bzw. ba^* das kleinste Element $x \in K$ mit

$$x = ax + b \qquad \text{bzw.} \qquad x = xa + b.$$

Beweis: Nach den Axiomen von Kozen ist a^*b das kleinste $x \in K$ mit $ax + b \leq x$. Jede Lösung von $x = ax + b$ in $\mathcal{K}$ erfüllt aber $ax + b \leq x$ und daher auch $x \geq a^*b$. Es bleibt zu zeigen, daß a^*b schon die Gleichung $ax + b = x$ erfüllt, wovon noch $a^*b \leq a(a^*b) + b$ fehlt. Hierin können wir $b = 1$ annehmen, so daß $a^* \leq aa^* + 1$ zu zeigen bleibt. Mit *KA-1* haben wir

$$\begin{array}{rcl} 1 + aa^* \leq a^* & \rightarrow & 1 \leq a^* \wedge aa^* \leq a^* \\ & \rightarrow & a \leq aa^* \leq a^* \\ & \rightarrow & a + aaa^* \leq aa^* + aa^* = aa^*. \\ & \rightarrow & 1 + a(1 + aa^*) \leq 1 + aa^*. \end{array}$$

Wegen *KA-2* gilt $\mathcal{K} \models \forall x(1 + ax \leq x \rightarrow a^* \leq x)$, womit wir $a^* \leq 1 + aa^*$ erhalten. Also hat jede lineare Gleichung $x = ax + b$ in $\mathcal{K}$ eine kleinste Lösung, eben a^*b. Entsprechend zeigt man die Aussage über ba^*. □

Zusammen mit den Schlußregeln der Logik bildet *KA* ein endliches und zur Herleitung gültiger Gleichungen zwischen regulären Ausdrücken korrektes und vollständiges Beweissystem.

Satz 3.8 (Vollständigkeitssatz, Kozen 1991) Für reguläre Ausdrücke r und s über $\Sigma = \{a_1, \ldots, a_n\}$ sind äquivalent:

(i) $L(r) = L(s)$.

(ii) In jeder Kleene'schen Algebra $\mathcal{K} = (K, +^K, 0^K, \cdot^K, 1^K, {}^{*K})$ gilt $\forall a_1 \ldots \forall a_n\, (r = s)$.

(iii) In der oben axiomatisierten Theorie KA ist $r = s$ beweisbar.

Beweis: iii) $\Longrightarrow$ i): Man hat zu zeigen, daß alle Axiome von KA in der Standardinterpretation durch Sprachen erfüllt sind. Wir zeigen das nur für *KA-1*, da es für *KA-2* genauso geht und für die übrigen Axiome recht einfach ist.

Sei $A \subseteq \Sigma^*$ ein beliebiges Element von 2^{Σ^*}. Dann ist

$$A^* \quad = \quad \bigcup_{k=0}^{\infty} A^k \quad = \quad \{\varepsilon\} \cup \bigcup_{k=1}^{\infty} A^k \quad = \quad \{\varepsilon\} \cup A \cdot (\bigcup_{k=0}^{\infty} A^k) \quad = \quad L(1) \cup A \cdot A^*.$$

Seien weiter $B, X \subseteq \Sigma^*$ mit $B \cup A \cdot X \subseteq X$. Dann ist (a) $B \subseteq X$ und (b) $AX \subseteq X$, und daher $A^k \cdot B \subseteq X$ für jedes k: $A^0 B = B \subseteq X$ gilt nach (a), und $A^{k+1}B = A(A^k B) \subseteq AX \subseteq X$ gilt mit Induktionsannahme und (b). Es folgt:

$$A^*B \quad = \quad (\bigcup_{k=0}^{\infty} A^k)B \quad = \quad \bigcup_{k=0}^{\infty} A^k B \quad \subseteq \quad X.$$

Damit ist das Axiom *KA-1* für die Sprachinterpretation gezeigt.

i) $\Longrightarrow$ ii): Das werden wir in Abschnitt 5.1 nachholen. (Der Beweis ist eine algebraische Formulierung der Argumentation mit minimalen deterministischen endlichen Automaten aus Abschnitt 4.) Mit 2.6 wurde ein Spezialfall dieser Behauptung gezeigt.

ii) $\Longrightarrow$ iii): Diese Behauptung folgt direkt aus dem Vollständigkeitssatz für die Prädikatenlogik. $\square$

Proposition 3.9 In jeder Kleene'schen Algebra ist * monoton: für $a \leq b$ ist $a^* \leq b^*$.

Beweis: Es ist zu zeigen, daß $a^* \leq (a+b)^*$ ist. Nach *KA-1* folgt das aus $ax + 1 \leq x$ für $x = (a+b)^*$. Wegen der Monotonie der Multiplikation ist aber

$$ax \leq (a+b)x = (a+b)(a+b)^* \leq (a+b)^* = x,$$

wobei benutzt wurde, daß nach *KA-1* für $y = (a+b)$ auch $yy^* \leq 1 + yy^* \leq y^*$ ist. $\square$

Proposition 3.10 In jeder Kleene'schen Algebra gilt $(a+b)^* = a^*(ba^*)^*$.

Beweis: Aus der Monotonie von * und $\cdot$ folgt

$$a^*(ba^*)^* \leq (a+b)^*((a+b)(a+b)^*)^* \leq (a+b)^*((a+b)^*)^* \leq ((a+b)^*)^*.$$

Aus der Transitivität von x^* (s. Aufgabe 3.2) folgt aber $(x^*)^* \leq x^*$. Für die Umkehrung benutzen wir $1+(a+b)x \leq x \to (a+b)^* \leq x$ mit $x = a^*(ba^*)^*$. Davon sind $1 \leq x$ und $bx = ba^*(ba^*)* \leq (ba^*)^* \leq x$ klar, und mit $aa^* \leq a^*$ auch $ax = aa^*(ba^*)^* \leq x$. □

Wir geben abschließend eine Kleene'sche Algebra an, die weder eine Sprach- noch eine Relationeninterpretation im bisherigen Sinne ist.

Beispiel 3.11 Seien Σ ein Alphabet und

$$\mathcal{L}_{\Sigma,\Sigma} := (2^{\Sigma^*\times\Sigma^*}, \cup, \emptyset, \cdot, \{(\varepsilon,\varepsilon)\}, ^*, \{(a,\varepsilon)\}, \{(\varepsilon,b)\})_{a,b\in\Sigma}$$

die Variante der Sprachinterpretation, bei der

$$A \cdot B := \{\, (a_1,a_2)\cdot(b_1,b_2) \mid (a_1,a_2) \in A, (b_1,b_2) \in B \,\}$$

aus der assoziativen Verknüpfung der Einfügung

$$(a_1,a_2)\cdot(b_1,b_2) := (a_1b_1, b_2a_2)$$

von Wortpaaren aus $\Sigma^* \times \Sigma^*$ entsteht, und

$$A^* := \bigcap \{\, B \subseteq \Sigma^* \times \Sigma^* \mid A \subseteq B,\, (\varepsilon,\varepsilon) \in B,\, B \cdot B \subseteq B \,\}$$

die iterierte Einfügung ist, d.h. der Abschluß von A unter Einfügung und 1.

Da + die Vereinigung von Relationen ist und $\cdot$ elementweise erklärt wurde, gelten die Eigenschaften eines idempotenten Halbrings. Wie bei der Sprachinterpretation sorgt die Wahl von A^* dafür, daß auch die anderen Axiome einer Kleene'schen Algebra erfüllt sind.

Unter der Algebra $Reg^{\mathcal{L}}_{\Sigma,\Sigma}$ der regulären Wortrelationen auf Σ verstehen wir die von den Relationen $\{(a,\varepsilon)\}$ und $\{(\varepsilon,a)\}$, mit $a \in \Sigma$, erzeugte reguläre Unteralgebra von $\mathcal{L}_{\Sigma,\Sigma}$.

3.4 Aktionslogik: die Axiomatisierung der Iteration * nach Pratt

Nach dem Ergebnis von Redko[22] kann es kein endliches Axiomensystem der Gleichungstheorie regulärer Algebren geben, das nur aus Gleichungen besteht. Erweitert man aber die regulären Operationen, so ist dies möglich, wie V. Pratt[21] gezeigt hat.

Als zusätzliche Operationen werden dabei die Rechts- und Linksdivision genommen, wie sie für Sprachen schon erwähnt wurden:

Definition 3.12 Eine AKTIONSALGEBRA ist eine Algebra $\mathcal{A} = (A, +, 0, \cdot, 1, ^*, \backslash, /)$, so daß gilt:

(i) $(A, \cdot, 1)$ ist ein Monoid, und $(A, +, 0)$ ist ein kommutatives, idempotentes Monoid.

(ii) Für alle $a, b, c \in A$ gilt: $a \le c/b \iff ab \le c \iff b \le a\backslash c$.

(iii) Für jedes $a \in A$ gilt: $1 + a^*a^* + a \;\le\; a^* \;\wedge\; \forall x(1 + xx + a \;\le\; x \implies a^* \le x)$.

Aus dem zweiten Axiom folgen die Distributiveigenschaften, z.B.

$$\begin{array}{rcccl} a(b+c) \le d & \iff & b + c \le a\backslash d & \iff & (b \le a\backslash d) \wedge (c \le a\backslash d) \\ & \iff & (ab \le d) \wedge (ac \le d) & \iff & ab + ac \le d. \end{array}$$

Daher ist $(A, +, 0, \cdot, 1)$ ein idempotenter Halbring. Das dritte Axiom formuliert direkt, daß a^* die reflexive transitive Hülle von a ist. Man kann damit z.B. zeigen, daß auch * eine monotone Operation ist.

Beispiel 3.13 (i) Die Sprachinterpretation $\mathcal{L}_\Sigma = (2^{\Sigma^*}, \cup, \emptyset, \cdot, \{\varepsilon\}, ^*, \{a\})_{a\in\Sigma}$ regulärer Ausdrücke wird durch Hinzunahme der Divisionen

$$A/B \;:=\; \{w \mid w \cdot B \subseteq A\}, \qquad \text{und} \qquad B\backslash A \;:=\; \{w \mid B \cdot w \subseteq A\}$$

zu einer Aktionsalgebra. Für $C, B \subseteq \Sigma^*$ ist nach der Definition C/B die größte Menge $X \subseteq \Sigma^*$ mit $XB \subseteq C$. Daher gilt dann $A \subseteq C/B \iff AB \subseteq (C/B)B \subseteq C$, und entsprechend für die andere Ungleichung in ii). Axiom iii) gilt, da L^* die kleinste Menge X mit $L \cup \{\varepsilon\} \subseteq X \subseteq \Sigma^*$ ist, die unter Verkettung abgeschlossen ist.

(ii) Jede Algebra $\mathcal{R} = (2^{M\times M}, \cup, \emptyset, \circ, 1_M, ^*)$ von zweistelligen Relationen auf einer Menge M kann zu einer Aktionsalgebra expandiert werden. Die Divisionen werden durch folgende Operationen interpretiert:

$$\begin{array}{rcl} B\backslash A & := & \{(m,n) \mid \forall x(B(x,m) \Rightarrow A(x,n))\}, \\ A/B & := & \{(m,n) \mid \forall x(B(n,x) \Rightarrow A(m,x))\}. \end{array}$$

Ähnlich wie bei der Sprachinterpretation ist hier A/B die größte Relation $X \subseteq M \times M$ mit $X \circ B \subseteq A$, und daher gilt das zweite Axiom. Das dritte Axiom ist klar, da A^* gerade als reflexive transitive Hülle von A definiert war.

Satz 3.14 (Pratt 1991) (i) Jede Aktionsalgebra wird durch Weglassen der Divisionen eine Kleene'sche Algebra.

(ii) Die Gleichungstheorie der Aktionsalgebren kann durch ein endliches System von Gleichungen vollständig axiomatisiert werden.

Beweis: (ii) Wir schreiben die Gleichungsaxiome mit Hilfe von Ungleichungen, mit $a \leq b$ statt $a + b = b$: Für alle Elemente soll gelten:

$$\begin{array}{rcll} a + (b + c) & = & (a + b) + c & (2) \\ a + b & = & b + a & (3) \\ a + a & = & a & (4) \\ a + 0 & = & a & (5) \end{array} \qquad \begin{array}{rcll} a\backslash b & \leq & a\backslash(b + b') & (6) \\ a(a\backslash b) \leq b & \leq & a\backslash(ab) & (7) \\ b/a & \leq & (b + b')/a & (8) \\ (b/a)a \leq b & \leq & (ba)/a & (9) \end{array}$$

$$\begin{array}{rcll} a(bc) & = & (ab)c & (10) \\ 1a = a & = & a1 & (11) \\ ab & \leq & (a + a')(b + b') & (12) \end{array} \qquad \begin{array}{rcll} 1 + a^*a^* + a & \leq & a^* & (13) \\ a^* & \leq & (a + b)^* & (14) \\ (a/a)^* & \leq & a/a & (15) \end{array}$$

Das letzte Axiom regelt die Beziehung zwischen der Iteration und der Division /.

Wir zeigen zuerst, daß alle diese (Un-)Gleichungen in einer Aktionsalgebra gelten.

Die Gleichungen (2) bis (5) und (10), (11) sind die Monoidforderungen aus (i). Die Monotonie der Multiplikation, (12), folgt aus den Distributiveigenschaften, die oben schon aus (ii) abgeleitet wurden. Die Ungleichungen in (7) ergeben sich aus (ii) durch

$$a\backslash b \leq a\backslash b \;\rightarrow\; a(a\backslash b) \leq b \qquad \text{und} \qquad ab \leq ab \;\rightarrow\; b \leq a\backslash(ab).$$

Ungleichung (6), die Monotonie von \ im zweiten Argument, folgt mit

$$\begin{array}{rll} a(a\backslash b) \leq b & \rightarrow\; a(a\backslash b) \leq (b + b') & \text{wegen } b \leq b + b' \\ & \rightarrow\; a\backslash b \leq a\backslash(b + b') & \text{nach (ii).} \end{array}$$

Analog erhält man (8) und (9). Ungleichung (13) ist Teil von (iii). Für (14), die Monotonie von *, ist zunächst

$$\begin{array}{rcl} 1 + (a + b)^*(a + b)^* + a & \leq & 1 + (a + b)^*(a + b)^* + (a + b) \\ & \leq & (a + b)^*, \quad \text{nach (iii)} \end{array}$$

und dann nach der Implikation aus (iii) auch $a^* \leq (a + b)^*$. Für Ungleichung (15) genügt es nach (iii),

$$1 \leq a/a, \quad (a/a)(a/a) \leq a/a, \quad \text{und} \quad a/a \leq a/a$$

zu zeigen, also daß a/a reflexiv und transitiv ist. Die Reflexivität ergibt sich aus (ii) wegen $1a \leq a \leftrightarrow 1 \leq a/a$, und die Transitivität durch

$$\begin{array}{rlll} a/a \leq a/a & \rightarrow & (a/a)a \leq a & \text{mit (ii)} \\ & \rightarrow & (a/a)(a/a)a \leq (a/a)a \leq a & \text{mit Monotonie von } \cdot \\ & \rightarrow & (a/a)(a/a) \leq a/a & \text{mit (ii).} \end{array}$$

Es bleibt zu zeigen, daß die Axiome einer Aktionsalgebra aus dem endlichen System von Gleichungen folgen. Das ist klar für die Monoidforderungen (i) und die Ungleichung $1 + a^*a^* + a \leq a^*$ aus (iii). Für die Äquivalenzen in (ii) argumentiert man wie folgt:

$$\begin{array}{lll} ab \leq c & \rightarrow \ a\backslash(ab) \leq a\backslash c & \text{wegen } ab \leq ab + c = c \text{ und } (6) \\ & \rightarrow \ b \leq a\backslash c & \text{wegen } (7) \end{array}$$

Umgekehrt ist

$$\begin{array}{lll} b \leq a\backslash c & \rightarrow \ ab \leq a(a\backslash c) & \text{mit } (12) \\ & \rightarrow \ ab \leq c & \text{nach } (7). \end{array}$$

Der andere Teil von (ii) geht entsprechend. Es bleibt die Implikation aus (iii) zu zeigen. Man hat

$$\begin{array}{lll} 1 + bb + a \leq b & \rightarrow \ 1 \leq b,\ bb \leq b,\ a \leq b & \\ & \rightarrow \ 1 \leq b,\ b \leq b/b,\ a \leq b & \text{mit (ii)} \\ & \rightarrow \ 1 \leq b,\ a \leq b/b & \\ & \rightarrow \ 1 \leq b,\ a^* \leq (b/b)^* \leq b/b & \text{nach } (14), (15) \\ & \rightarrow \ 1 \leq b,\ a^*b \leq b & \text{mit (ii)} \\ & \rightarrow \ a^* \leq a^*b,\ a^*b \leq b & \text{mit } (12) \\ & \rightarrow \ a^* \leq b. & \end{array}$$

Beachte, daß wir neben (15) ebenso $(a\backslash a)^* \leq a\backslash a$ aus den Axiomen der Aktionsalgebra erhalten, also nicht eigens im Gleichungssystem fordern müssen.

(i) Da die Distributiveigenschaften aus (ii) folgen, sind nur noch die Axiome *KA-1* und *KA-2* aus denen der Aktionsalgebra herzuleiten.

$$\begin{array}{lll} 1 + a^*a^* + a \leq a^* & \rightarrow \ 1 + a^*a^* \leq a^*,\ a \leq a^* & \\ & \rightarrow \ 1 + aa^* \leq 1 + a^*a^* & \text{wegen } (12) \\ & \rightarrow \ 1 + aa^* \leq 1 + a^*a^* + a \leq a^*. & \end{array}$$

Den zweiten Teil von *KA-1* erhalten wir durch

$$\begin{array}{lll} b + ax \leq x & \rightarrow \ b \leq x,\ ax \leq x & \\ & \rightarrow \ b \leq x,\ a \leq x/x & \text{mit (ii)} \\ & \rightarrow \ b \leq x,\ 1 + (x/x)^*(x/x)^* + a \leq (x/x)^* & \text{mit } (13) \\ & \rightarrow \ b \leq x,\ a^* \leq x/x & \text{mit (iii) und } (15) \\ & \rightarrow \ b \leq x,\ a^*x \leq x & \text{nach } (7) \\ & \rightarrow \ a^*b \leq a^*x \leq x & \text{mit } (12). \end{array}$$

Entsprechend beweist man *KA-2*. □

Korollar 3.15 Für reguläre Ausdrücke r und s ist die Gleichung $r = s$ in der Aktionsalgebra genau dann beweisbar, wenn $L(r) = L(s)$.

Das ergibt sich aus (i) mit dem Vollständigkeitssatz von Kozen. Das Korollar kann aber nicht verallgemeinert werden für Gleichungen, in denen die Divisionen vorkommen, oder für Aussagen, die logische Beziehungen zwischen Gleichungen formulieren. Siehe dazu Aufgabe 3.4.

Der Name Aktions*logik* rührt daher, daß man die Operatoren $\cdot$, $\backslash$ und $/$ als zeitlich gerichtete logische Verknüpfungen von Aussagen (über Handlungen) interpretieren kann:

$$\begin{array}{rcccl} & & a \cdot b & \simeq & a \text{ und dann } b \\ a \leftarrow b & := & a/b & \simeq & a, \text{ falls danach } b \\ b \rightarrow a & := & b\backslash a & \simeq & a, \text{ falls davor } b. \end{array}$$

Die Iteration wird in der Aktionsalgebra als eine Form der Induktion aufgefaßt. Das Induktionsaxiom

$$A(0) \wedge \forall n\,(A(n) \rightarrow A(n+1)) \Longrightarrow \forall n\, A(n)$$

in der Arithmetik unterscheidet zwischen Formeln A und Objekten $n \in I\!\!N$. Der Unterschied verschwindet in der Aktionslogik, wodurch die Induktion zu einer Iteration von $A \rightarrow A$ wird und

$$a \cdot (a \rightarrow a)^* \leq a$$

oder das gleichwertige $(a\backslash a)^* \leq (a\backslash a)$ übrig bleibt. Pratt nennt es das Axiom der REINEN (d.h. einsortigen) INDUKTION.

Aufgaben

Aufgabe 3.1 Das Lemma von Arden besagt, daß die Aussage

$$\forall a \forall b\ (1 \not\leq a \rightarrow \forall x(x = ax + b \rightarrow x = a^* b)) \tag{16}$$

in $\mathcal{L}_\Sigma$ gilt.

(i) Zeige, daß die Voraussetzung $1 \not\leq a$ notwendig ist.

(ii) Zeige, daß die Aussage (16) nicht in jeder Kleene'schen Algebra gilt.

Hinweise: (i) Gib für $\Sigma = \{a, b\}$ Sprachen $A, B \subseteq \Sigma^*$ mit $\varepsilon \in A$ an, für die die Gleichung $X = AX \cup B$ keine eindeutig bestimmte Lösung $X \subseteq \Sigma^*$ hat. (ii) Benutze eine Relationenalgebra $\mathcal{R}_M$ mit $|M| = 2$ und geeigneten Relationen $A, X \subseteq M \times M$ mit $A \circ X = X$.

Aufgabe 3.2 a) Zeige, daß die Conway'sche Unstetigkeit $\mathcal{M}$ keine Kleene'sche Algebra ist.

b) Beweise in KA (mit den Definitionen aus dem Beweis von Satz 3.5):

(i)	a^* ist reflexiv und transitiv,	(iv)	$(cd)^*c = c(dc)^*$
(ii)	Ist $x \geq a$ reflexiv und transitiv, so ist $a^* \leq x$,	(v)	$a^*b = a(a^*b) + b$,
(iii)	$\forall x(ax = xb \;\Rightarrow\; a^*x = xb^*)$,	(vi)	$0^* = 1 = 1^*$.

Aufgabe 3.3 (i) Sei $\mathcal{K} = (K, +, 0, \cdot, 1, ^*)$ eine Kleene'sche Algebra. Bilde mit der durch $a \cdot' b := b \cdot a$ definierten Operation die Algebra $\mathcal{K}' := (K, +, 0, \cdot', 1, ^*)$. Zeige, daß auch $\mathcal{K}'$ eine Kleene'sche Algebra ist, wobei man * nicht zu ändern braucht.

(ii) Sei $\mathcal{K}_i = (K_i, +_i, 0_i, \cdot_i, 1_i, ^{*_i})$ für $i = 1, 2$ eine Kleene'sche Algebra, und $\mathcal{K}_1 \times \mathcal{K}_2 := (K, +, 0, \cdot, 1, ^*)$ definiert durch

$$\begin{array}{rcl@{\qquad}rcl}
K & := & K_1 \times K_2, & (a_1, a_2) + (b_1, b_2) & := & (a_1 +_1 b_1, a_2 +_2 b_2) \\
0 & := & (0_1, 0_2), & (a_1, a_2) \cdot (b_1, b_2) & := & (a_1 \cdot_1 b_1, a_2 \cdot_2 b_2) \\
1 & := & (1_1, 1_2), & (a_1, a_2)^* & := & (a_1^{*_1}, a_2^{*_2})
\end{array}$$

Zeige, daß $\mathcal{K}_1 \times \mathcal{K}_2$ eine Kleene'sche Algebra ist.

Aufgabe 3.4 (i) Seien $a \neq b$ Buchstaben aus Σ. Dann gilt $L(a/b) = L(0)$, aber $a/b = 0$ ist in der Theorie der Aktionsalgebren nicht beweisbar.

(ii) Sprach- und Relationeninterpretation sind nach Beispiel 3.13 Aktionsalgebren. Bei der Sprachinterpretation gilt die Aussage

$$\forall x \; \forall y \; (1 \leq xy \;\to\; 1 \leq x \wedge 1 \leq y).$$

Zeige, daß diese Aussage nicht in jeder Relationenalgebra gilt.

(iii) Gilt in KA oder ACT, der Aktionsalgebra, die Aussage $\forall a \forall y (1 \leq a^*y \to 1 \leq y)$?

(iv) Wie sind die zur Einfügung passenden Divisionen $/$ und $\backslash$ zu definieren, damit aus der Kleene'schen Algebra $\mathcal{L}_{\Sigma\times\Sigma}$ aus Beispiel 3.11 eine Aktionsalgebra wird?

4 Spracherkennung mit endlichen Automaten

In manchen Fällen, insbesondere bei der lexikalischen Analyse von Programmen, ist man gar nicht an den syntaktischen Strukturen eines $w \in \Sigma^*$ bezüglich einer Sprache $L \subseteq \Sigma^*$ interessiert, sondern nur daran, *ob* w in L liegt oder nicht.

Das ist das Problem der SPRACHERKENNUNG, das wir im Folgenden für reguläre Sprachen untersuchen. Die Aufgabe besteht also darin, zu einem regulären Ausdruck r über Σ ein Verfahren $\mathcal{A}_r$ anzugeben, mit dem für jedes $w \in \Sigma^*$ festgestellt werden kann, ob $w \in L(r)$. Dazu verwenden wir Automaten mit endlichem Gedächtnis, die beliebig lange Wörter $w \in \Sigma^*$ lesen und zwischen akzeptablen und unakzeptablen Wörtern unterscheiden können.

4.1 Endliche Automaten

Ein endlicher (erkennender) Automat liest ein Wort Buchstabe für Buchstabe, und merkt sich dabei endlich viel Information in maschineninternen Zuständen, seinem Arbeitsspeicher. Das Lesen eines Buchstabens bewirkt eine Änderung der gespeicherten Information. Wenn man nicht verlangt, daß der Folgezustand eindeutig besimmt ist, entspricht dem Lesen eines Buchstabens also eine zweistellige Relation zwischen Zuständen.

Definition 4.1 Ein ENDLICHER AUTOMAT $\mathcal{A} = (Q, \vdash_a, I, F)_{a\in\Sigma}$ besteht aus

(i) endlichen Mengen Q, den ZUSTÄNDEN, und Σ, dem EINGABEALPHABET,

(ii) zweistelligen Relationen $\vdash_a \subseteq Q \times Q$ für jedes $a \in \Sigma$, den ÜBERGANGSRELATIONEN,

(iii) ausgezeichneten Teilmengen $I, F \subseteq Q$, den ANFANGS- und den ENDZUSTÄNDEN.

Aus den Relationen $\vdash_a$ erhält man für jedes Wort $w \in \Sigma^*$ eine Relation $\vdash_w$ durch

$$\vdash_\varepsilon := \{ (q,q) \mid q \in Q \}, \qquad \vdash_{va} := \vdash_v \circ \vdash_a \qquad \text{für } v \in \Sigma^* \text{ und } a \in \Sigma.$$

$\mathcal{A}$ ERKENNT $w \in \Sigma^*$, falls $p \vdash_w q$ für einen Anfangszustand $p \in I$ und einen Endzustand $q \in F$. DIE VON $\mathcal{A}$ ERKANNTE SPRACHE ist

$$L(\mathcal{A}) := \bigcup_{p\in I,\, q\in F} \{ w \in \Sigma^* \mid p \vdash_w q \}.$$

Zwei Automaten heißen (SPRACH-) ÄQUIVALENT, wenn sie dieselbe Sprache erkennen.

Manchmal faßt man die Übergangsrelationen $\vdash_a$ zu einer Relation

$$\delta \;:=\; \{\, (p,a,q) \;|\; p \vdash_a q \,\} \;\subseteq Q \times \Sigma \times Q$$

zusammen und schreibt $\mathcal{A} = (Q, \Sigma, \delta, I, F)$.

Ein Automat $\mathcal{A}$ heißt (vollständig) DETERMINISTISCH, wenn es genau einen Anfangszustand gibt und die Übergangsrelationen $\vdash_a$ (totale) Funktionen sind, d.h. zu $p \in Q$ und $a \in \Sigma$ (mindestens und) höchstens ein $q \in Q$ mit $p \vdash_a q$ existiert. Wenn $\mathcal{A}$ deterministisch ist, d.h. $I = \{q_0\}$ und δ eine Funktion $\delta : Q \times \Sigma \to Q$ ist, schreibt man meist $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ und setzt δ durch

$$\delta(q, \varepsilon) := q, \quad \delta(q, aw) := \delta(\delta(q,a), w).$$

zu einer auf ganz $Q \times \Sigma^*$ definierten Funktion δ fort. Die von $\mathcal{A}$ erkannte Sprache ist dann

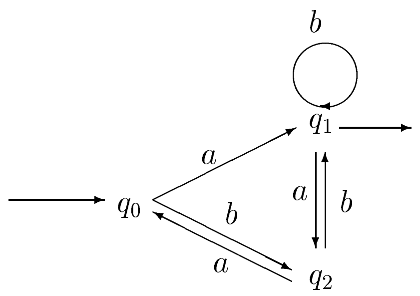
$$L(\mathcal{A}) = \{\, w \in \Sigma^* \;|\; \delta(q_0, w) \in F \,\}.$$

Beispiel 4.2 Sei $\mathcal{A} = (Q, \vdash_a, \vdash_b, I, F)$ der endliche Automat mit $Q = \{q_0, q_1, q_2\}$, $I = \{q_0\}$, $F = \{q_1\}$,

$$\vdash_a \;=\; \{(q_0,q_1),(q_1,q_2),(q_2,q_0)\} \quad \text{und} \quad \vdash_b \;=\; \{(q_0,q_2),(q_1,q_1),(q_2,q_1)\}.$$

$\mathcal{A}$ ist deterministisch und akzeptiert beispielsweise die Wörter ab^n und $(ba)^n a(ab)^m$.

(i) Man kann $\mathcal{A}$ durch einen (Multi-) GRAPHEN darstellen, wobei die Relationen $\vdash_a$ und $\vdash_b$ als entsprechend markierte Kanten erscheinen:



(ii) Man kann $\mathcal{A}$ auch durch seine Übergangsfunktion δ als MASCHINENTAFEL angeben:

		a	b
$\to$	q_0	q_1	q_2
$\leftarrow$	q_1	q_2	q_1
	q_2	q_0	q_1

(iii) Die Maschinentafel kann man auch als ein GO-TO-Programm auffassen, wobei die Zustände die Zeilennummern des Programms sind:

$$\begin{array}{rl} \to\ q_0: & \textit{if input} = a \textit{ go to } q_1; \quad \textit{if input} = b \textit{ go to } q_2 \\ q_1: & \textit{if input} = a \textit{ go to } q_2; \quad \textit{if input} = b \textit{ go to } q_1 \\ q_2: & \textit{if input} = a \textit{ go to } q_0; \quad \textit{if input} = b \textit{ go to } q_1 \end{array}$$

(iv) In Abschnitt 5.1 werden wir endliche Automaten als Matrizen über einer Kleene'schen Algebra $\mathcal{K}$ darstellen, wobei man die Zeilen- und Spaltenindizes als Zustände auffaßt und Elemente von K als Einträge zuläßt.

Nicht-deterministische endliche Automaten, die eine gegebene Sprache erkennen, sind oft leichter zu finden als deterministische, da sie eng mit regulären Ausdrücken zusammenhängen. Das werden wir im nächsten Abschnitt sehen.

Vorher zeigen wir, daß jeder nichtdeterministische endliche Automat durch einen deterministischen simuliert werden kann. Alle Rechnungen, die ein nichtdeterministischer endlicher Automat (als von einander unabhängige Alternativen) ausführen kann, kann man mit einem deterministischen endlichen Automaten "parallel" simulieren. Dazu merkt man sich im Speicherzustand der deterministischen Maschine die Menge der von der nichtdeterministischen Maschine erreichbaren Zustände. Die neue Zustandsmenge ist die Potenzmenge der vorherigen, also wiederum endlich.

Satz 4.3 Zu jedem endlichen Automaten $\mathcal{A}$ über Σ gibt es einen deterministischen endlichen Automaten $\mathcal{A}'$ über Σ mit $L(\mathcal{A}) = L(\mathcal{A}')$.

Beweis: Sei $\mathcal{A} = (Q, \vdash_a, I, F)_{a\in\Sigma}$ ein nicht-deterministischer endlicher Automat. Konstruiere den "Potenzmengenautomaten" $\mathcal{A}' := (Q', \Sigma, \delta', q_0', F')$ durch

$$\begin{array}{rcllrcl} Q' & := & \{\, B \mid B \subseteq Q \,\}, & \qquad & \delta'(B,a) & := & \bigcup_{p\in B}\{\, q \in Q \mid p \vdash_a q \,\}, \\ q_0' & := & I, & & F' & := & \{\, B \subseteq Q \mid B \cap F \neq \emptyset \,\}. \end{array}$$

Offensichtlich ist $\mathcal{A}'$ ein deterministischer endlicher Automat über Σ.

$L(\mathcal{A}) \subseteq L(\mathcal{A}')$: Sei $a_1 \cdots a_n \in L(\mathcal{A})$. Dann gibt es $q_0, \ldots, q_n \in Q$ mit $q_0 \in I, q_n \in F$ und

$$q_0 \vdash_{a_1} q_1 \vdash_{a_2} q_2 \vdash_{a_3} \ \ldots\ q_{n-1} \vdash_{a_n} q_n.$$

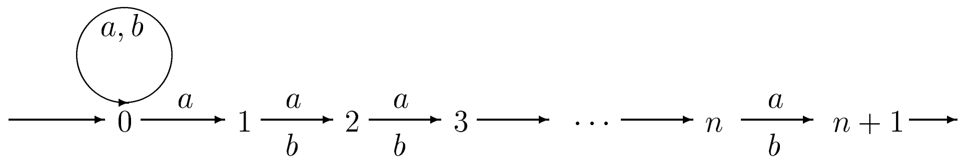
Für $B_0 := \{q_0\}$ und $B_i := \delta(B_{i-1}, a_i)$ für $i = 1, \ldots, n$ ist jedes $q_i \in B_i \in Q'$ und $q_n \in F \cap B_n$. Also ist $\delta'(q_0', a_1 \cdots a_n) = B_n \in F'$, d.h. $a_1 \cdots a_n \in L(\mathcal{A}')$.

$L(\mathcal{A}') \subseteq L(\mathcal{A})$: Sei $a_1 \cdots a_n \in L(\mathcal{A}')$. Dann gibt es $B_0, \ldots, B_n \subseteq Q$ mit $B_0 = q_0' = I$, $B_i = \delta'(B_{i-1}, a_i)$ für $i = 1, \ldots, n$ und $B_n \cap F \neq \emptyset$. Wähle $q_n \in B_n \cap F$. Ist für ein $i \leq n$

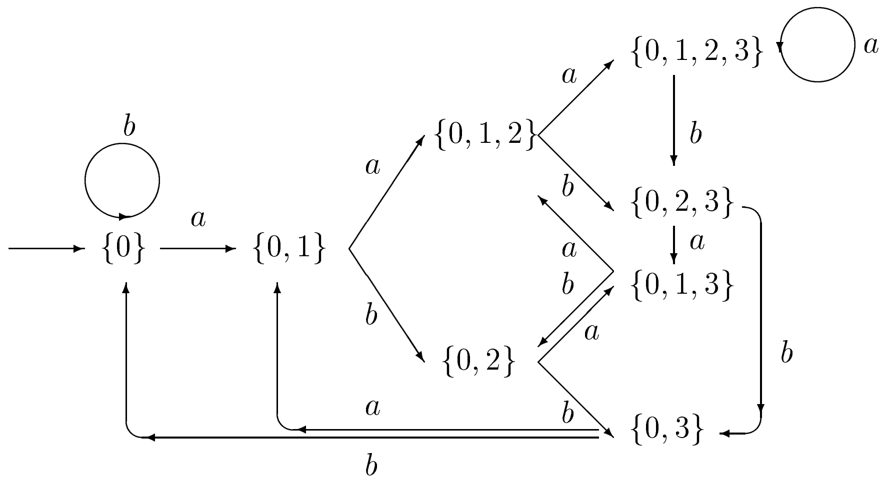
schon $q_i \in B_i$ gewählt, so gibt es wegen $B_i = \delta'(B_{i-1}, a_i) = \bigcup_{p \in B_{i-1}} \{ q \in Q \mid p \vdash_{a_i} q \}$ ein $q_{i-1} \in B_{i-1}$ mit $q_{i-1} \vdash_{a_i} q_i$. Wegen $B_0 = I$ ist q_0 ein Anfangszustand von $\mathcal{A}$, und wegen $q_n \in F$ ist mit $q_0 \vdash_{a_1} q_1 \vdash_{a_2} \dots q_{n-1} \vdash_{a_n} q_n$ die Behauptung $a_1 \cdots a_n \in L(\mathcal{A})$ gezeigt. $\square$

Die Anzahl der Zustände von $\mathcal{A}'$ wächst im Allgemeinen bis zu $2^{|Q|}$. Wir werden später sehen, daß man zu $\mathcal{A}'$ stets einen deterministischen Automaten mit minimaler Zustandszahl finden kann, der $L(\mathcal{A}')$ erkennt. Aber auch dieser minimierte Automat kann noch $O(2^{|Q|})$ viele Zustände haben.

Beispiel 4.4 Die durch den Ausdruck $(a+b)^*a(a+b)^n$ definierte Sprache wird von



erkannt, einem nicht-deterministischen Automaten mit $n+2$ Zuständen. Die Potenzmengenkonstruktion führt zu einem binären Baum der Tiefe $n+1$ (hier für $n = 2$), dessen Blätter die Endzustände sind:



Ob ein Wort akzeptiert wird, hängt davon ab, ob sein drittletzter (bzw. $n+1$-letzter) Buchstabe ein a ist. In den akzeptierenden Zuständen $\{0,1,2,3\}$ bis $\{0,3\}$ der letzten Spalte merkt man sich, ob die letzten drei Buchstaben eines gelesenen Wortes *aaa*, *aab*, *aba*, oder *abb* waren. Man muß hier auch nach den vorletzten und letzten Buchstaben unterscheiden, da diese bei Verlängerung des Eingabewortes um einen bzw. zwei Buchstaben benötigt werden, um die Zugehörigkeit des verlängerten Eingabeworts zur Sprache feststellen zu können. Jeder die Sprache erkennende deterministische Automat braucht also 2^{n+1} verschiedene Endzustände.

4.2 Reguläre Ausdrücke und endliche Automaten

Wir zeigen jetzt, daß eine Sprache $L \subseteq \Sigma^*$ genau dann durch einen regulären Ausdruck über Σ definierbar ist, wenn sie von einem endlichen Automaten über Σ akzeptiert wird.

Satz 4.5 (S.C.Kleene 1956)

(i) Zu jedem regulären Ausdruck r über Σ gibt es einen endlichen Automaten $\mathcal{A}$ über Σ, so daß $L(r) = L(\mathcal{A})$.

(ii) Zu jedem endlichen Automaten Automaten $\mathcal{A}$ über Σ gibt es einen regulären Ausdruck r über Σ mit $L(r) = L(\mathcal{A})$.

Beweis: i) Zu zwei endlichen Automaten $\mathcal{A}_i = (Q_i, \vdash_{i,a}, I_i, F_i)_{a\in\Sigma}$ mit o.B.d.A. disjunkten Mengen Q_1 und Q_2 konstruieren wir zunächst folgende neuen Automaten:

(i) Die Parallelschaltung $\mathcal{A}_1 + \mathcal{A}_2$ von $\mathcal{A}_1$ und $\mathcal{A}_2$ sei der endliche Automat

$$(Q_1 \cup Q_2, \vdash_{1,a} \cup \vdash_{2,a}, I_1 \cup I_2, F_1 \cup F_2)_{a\in\Sigma}.$$

(ii) Die Hintereinanderschaltung $\mathcal{A}_1 \cdot \mathcal{A}_2$ von $\mathcal{A}_1$ und $\mathcal{A}_2$ sei der endliche Automat $(Q_1 \cup Q_2, \vdash_a, I, F_2)_{a\in\Sigma}$ mit

$$\begin{aligned} I &:= \begin{cases} I_1, & \text{falls } I_1 \cap F_1 = \emptyset, \\ I_1 \cup I_2, & \text{sonst,} \end{cases} \\ \vdash_a &:= \vdash_{1,a} \cup \vdash_{2,a} \cup \{ (p,q) \mid p \in Q_1, r \in F_1, q \in I_2,\ p \vdash_{1,a} r \}. \end{aligned}$$

(iii) Die Rückkopplung $\mathcal{A}_1^*$ von $\mathcal{A}_1$ sei der endliche Automat $(Q, \vdash_a, I, F)_{a\in\Sigma}$ mit

$$\begin{aligned} Q &:= \begin{cases} Q_1, & \text{falls } I_1 \cap F_1 \neq \emptyset, \\ Q_1 \cup \{q_0\} & \text{sonst, mit einem neuen Zustand } q_0 \notin Q_1, \end{cases} \\ \vdash_a &:= \vdash_{1,a} \cup \{ (p,q) \mid p \in Q_1, q \in I_1, r \in F_1,\ p \vdash_{1,a} r \} \\ I &:= \begin{cases} I_1, & \text{falls } I_1 \cap F_1 \neq \emptyset, \\ I_1 \cup \{q_0\}, & \text{sonst,} \end{cases} \\ F &:= \begin{cases} F_1, & \text{falls } I_1 \cap F_1 \neq \emptyset, \\ F_1 \cup \{q_0\}, & \text{sonst.} \end{cases} \end{aligned}$$

Diese Konstruktionen entsprechen den Zusammenschaltungen der Flußdiagramme $P(r)$ aus Abschnitt 2.3. (Die Verwendung von mit ε markierten Kanten wurde vermieden, damit wieder $\vdash_1 = 1_Q$ gilt.) Man überlegt sich leicht, daß folgende Beziehungen gelten:

$$\begin{aligned} L(\mathcal{A}_1 + \mathcal{A}_2) &= L(\mathcal{A}_1) \cup L(\mathcal{A}_2), \\ L(\mathcal{A}_1 \cdot \mathcal{A}_2) &= L(\mathcal{A}_1) \cdot L(\mathcal{A}_2), \\ L(\mathcal{A}_1^*) &= L(\mathcal{A}_1)^*. \end{aligned} \tag{17}$$

Nun konstruieren wir zu regulären Ausdrücken r endliche Automaten $\mathcal{A}_r$ wie folgt:

$$\begin{array}{lll}
\mathcal{A}_0 & = & (\emptyset, \vdash_a, \emptyset, \emptyset)_{a\in\Sigma}, \quad \text{mit} \quad \vdash_a \; = \; \emptyset \\
\mathcal{A}_1 & = & (\{q\}, \vdash_a, \{q\}, \{q\})_{a\in\Sigma}, \quad \text{mit} \quad \vdash_a \; = \; \emptyset \\
\mathcal{A}_b & = & (\{q_0, q_1\}, \vdash_a, \{q_0\}, \{q_1\})_{a\in\Sigma}, \quad \text{mit} \quad \vdash_a \; = \; \begin{cases} \{(q_0, q_1)\}, & \text{falls } b \equiv a, \\ \emptyset, & \text{sonst.} \end{cases} \\
\mathcal{A}_{(r+s)} & = & \mathcal{A}_r + \mathcal{A}_s \\
\mathcal{A}_{(r\cdot s)} & = & \mathcal{A}_r \cdot \mathcal{A}_s \\
\mathcal{A}_{r^*} & = & \mathcal{A}_r^*
\end{array}$$

Für die atomaren Ausdrücke r ist offenbar $L(\mathcal{A}_r) = L(r)$, und für die übrigen zeigt man es induktiv mit Hilfe von (17).

ii) Sei $\mathcal{A} = (Q, \vdash_a, I, F)_{a\in\Sigma}$ mit $Q = \{x_0, \ldots, x_n\}$. Wir fassen die Zustände als Variable auf und den Automaten als ein Gleichungssystem. Dazu ordnen wir dem Zustand x_i die Sprache

$$L(x_i) := \bigcup_{x\in F} \{\, w \in \Sigma^* \mid x_i \vdash_w x \,\}$$

zu. Sei $r_{i,j}$ ein variablenfreier regulärer Ausdruck mit $L(r_{i,j}) = \{\, a \in \Sigma \mid x_i \vdash_a x_j \,\} \subseteq \Sigma$, so daß

$$L(x_i) = \begin{cases} \bigcup_{j=0}^n L(r_{i,j} \cdot x_j) & \text{falls } x_i \notin F \\ \bigcup_{j=0}^n L(r_{i,j} \cdot x_j) \cup \{\varepsilon\} & \text{falls } x_i \in F. \end{cases}$$

Daher bilden die $L(x_i)$ eine Lösung der Gleichungen

$$\begin{array}{lll}
x_0 & = & r_1(x_0, \ldots, x_n) \\
\vdots & \vdots & \quad\vdots \\
x_n & = & r_n(x_0, \ldots, x_n)
\end{array}$$

mit den regulären Ausdrücken

$$r_i(x_0, \ldots, x_n) := \begin{cases} r_{i,0} \cdot x_0 + \ldots + r_{i,n} \cdot x_n, & \text{falls } x_i \notin F \\ r_{i,0} \cdot x_0 + \ldots + r_{i,n} \cdot x_n + 1, & \text{sonst.} \end{cases} \tag{18}$$

Mit den Halbringeigenschaften bringen wir r_n auf die Form

$$r_n(x_0, \ldots, x_n) = s_n(x_0, \ldots, x_{n-1}) + r_{n,n} \cdot x_n.$$

Nach dem Lemma von Arden ist die Gleichung $x_n = s_n(x_0, \ldots, x_{n-1}) + r_{n,n} \cdot x_n$ zu

$$\begin{array}{lll}
x_n & = & r_{n,n}^* \cdot s_n(x_0, \ldots, x_{n-1}) \\
 & = & r_{n,n}^* \cdot r_{n,0} \cdot x_0 + \ldots + r_{n,n}^* \cdot r_{n,n-1} \cdot x_{n-1} \quad (\text{ggf. } + r_{n,n}^*)
\end{array} \tag{19}$$

äquivalent, da $\varepsilon \notin L(r_{n,n})$. Ersetze x_n in allen Gleichungen durch den Ausdruck (19) ohne x_n, so daß nach Weglassen der n-ten Gleichung ein neues System

$$\begin{array}{lllll}
x_0 & = & r_1'(x_0, \ldots, x_{n-1}) & = & r_{0,0}' \cdot x_0 + \ldots + r_{0,n-1}' \cdot x_{n-1} \; (+1) \\
\vdots & \vdots & \quad\vdots & \vdots & \quad\vdots \\
x_{n-1} & = & r_{n-1}'(x_0, \ldots, x_{n-1}) & = & r_{n-1,0}' \cdot x_0 + \ldots + r_{n-1,n-1}' \cdot x_{n-1} \; (+1)
\end{array}$$

übrigbleibt, das dieselben Sprachen $L(x_i)$ für $i = 0, \ldots, n-1$ definiert. Beachte, daß wegen der Form von (18) und (19) wieder Koeffiziententerme $r'_{i,j}$ mit $\varepsilon \notin L(r'_{i,j})$ entstehen. Daher können wir auf dieselbe Weise $x_{n-1}, \ldots, x_0$ eliminieren, so daß wir schließlich einen variablenfreien regulären Ausdruck r_0 über Σ erhalten mit $L(x_0) = L(r_0)$. Bilden wir auf dieselbe Weise reguläre Ausdrücke für jede Variable in $I = \{x_{i_1}, \ldots, x_{i_k}\}$, so haben wir $L(\mathcal{A}) = L(x_{i_1}) \cup \ldots \cup L(x_{i_k}) = L(r_{i_1} + \ldots + r_{i_k})$. □

Beispiel 4.6 Sei $\mathcal{A}$ der Automat aus Beispiel 4.2. Sein Gleichungssystem ist

$$\begin{array}{rcllcl} x_0 & = & 0x_0 + ax_1 + bx_2 & & = & ax_1 + bx_2 \\ x_1 & = & 0x_0 + bx_1 + ax_2 + 1 & & = & bx_1 + ax_2 + 1 \\ x_2 & = & ax_0 + bx_1 + 0x_2 & & = & ax_0 + bx_1 \end{array}$$

Der Lösungsausdruck für x_2 ist $x_2 = (ax_0 + bx_1) + 0x_2 = 0^*(ax_0 + bx_1) = ax_0 + bx_1$. Setzt man ihn für x_2 in den ersten beiden Gleichungen ein, so erhält man

$$\begin{array}{rclcl} x_0 & = & ax_1 + b(ax_0 + bx_1) & = & bax_0 + (a + bb)x_1 \\ x_1 & = & bx_1 + a(ax_0 + bx_1) + 1 & = & aax_0 + (b + ab)x_1 + 1. \end{array}$$

Der Lösungsausdruck für x_1 wird $x_1 = (aax_0 + 1) + (b + ab)x_1 = (b + ab)^*(aax_0 + 1)$, was in die erste Gleichung eingesetzt zu

$$\begin{array}{rcl} x_0 & = & bax_0 + (a + bb)((b + ab)^*(aax_0 + 1)) \\ & = & (ba + (a + bb)(b + ab)^*aa)x_0 + (a + bb)(b + ab)^* \end{array}$$

führt. Der reguläre Ausdruck für die Sprache $L(\mathcal{A})$ ist der Lösungsausdruck für x_0, also $x_0 = (ba + (a + bb)(b + ab)^*aa)^*(a + bb)(b + ab)^*$.

Der Beweis zeigte, daß man endliche Automaten als Gleichungssysteme verstehen kann, deren rechte Seiten Summen von Ausdrücken $0x$, ax mit $a \in \Sigma$ oder 1 sind. Reguläre Ausdrücke bezeichnen die Komponenten der Lösungen solcher Gleichungen.

Wir wollen uns von der Einschränkung, daß keine Summanden der Form x auftreten, frei machen. Allerdings geht dabei die Eindeutigkeit der Lösung verloren: $x = ax + x$ etwa hat unendlich viele Lösungen, z.B. 0 und jedes a^*A für $A \subseteq \Sigma^*$. Um auf den rechten Seiten der Gleichungen allgemeine Linearkombinationen der Variablen erlauben zu können, muß man sich mit Lemma 3.7 auf die kleinsten Lösungen beschränken:

Definition 4.7 Ein RECHTSLINEARES GLEICHUNGSSYSTEM über Σ ist ein System E von Gleichungen für paarweise verschiedene REKURSIONSVARIABLE $x_1, \ldots, x_m$,

$$\begin{array}{rcl} x_1 & = & r_1(x_1, \ldots, x_m) \\ \vdots & \vdots & \vdots \\ x_m & = & r_m(x_1, \ldots, x_m), \end{array} \tag{20}$$

dessen rechte Seiten r_i Summen von Wörtern w oder ux sind, wobei $u, w \in \Sigma^+ \cup \{0, 1\}$ und $x \in \{x_1, \ldots, x_m\}$ ist. Eine Sprache $L \subseteq \Sigma^*$ heißt RECHTSLINEAR, wenn sie eine Komponente der kleinsten Lösung eines rechtslinearen Gleichungssystems über Σ ist.

Satz 4.8 Eine Sprache ist genau dann rechtslinear, wenn sie regulär ist.

Beweis: $\Longleftarrow$: Sei $\mathcal{A} = (\{x_1, \ldots, x_n\}, \vdash_a, I, F)_{a\in\Sigma}$ ein endlicher Automat mit $L = L(\mathcal{A})$. Nach dem vorigen Beweis kann $\mathcal{A}$ durch ein (eindeutig lösbares) rechtslineares Gleichungssystem dargestellt werden, dessen Variable x_i die Zustände von $\mathcal{A}$ sind, so daß $L = \bigcup_{i\in I} L(x_i)$ für die Lösung mit den Komponenten $L(x_1), \ldots, L(x_n)$. Durch Hinzunahme einer Gleichung $x_0 = \Sigma_{i\in I}\, x_i$ folgt die Behauptung.

$\Longrightarrow$: Es genügt, das rechtslineare Gleichungssystem in die Darstellung eines endlichen Automaten umzuformen, ohne daß sich die kleinste Lösung ändert:

Beh.: Jedes rechtslineare Gleichungssystem E, dessen kleinste Lösung in den Komponenten $x_1, \ldots, x_k$ paarweise verschieden ist, läßt sich in ein rechtslineares System E umformen, dessen Summanden von der Form 0,1 oder ax mit $a \in \Sigma$ sind und dessen kleinste Lösung in den Komponenten $x_1, \ldots, x_k$ mit der kleinsten Lösung von E übereinstimmt.

(a) Beseitige zuerst alle Summanden x. Konstruiere dazu auf der Menge X der Variablen den Graphen $(X, \to^*)$, wobei $\to^*$ die reflexive transitive Hülle der Relation

$$x_i \to x_j :\Longleftrightarrow x_j \text{ ist Summand in der Gleichung } x_i = r_i$$

ist. Sind $x_i \to^* x_j$ und $x_j \to^* x_i$, so ist $L(x_i) = L(x_j)$ in jeder Lösung von (20). Man bilde r'_j aus der Gleichung $x_j = r_j$, indem man in r_j alle Summanden x_i und x_j durch 0 ersetzt. Ersetze die Summanden x_j in allen übrigen Gleichungen durch r'_j und Summanden ax_j durch ax_i. Für $i \neq j$ entferne $x_j = r_j$. Im neuen Gleichungssystem tritt x_j nicht mehr als Summand auf. (Summanden 0 kann man bei $j = i < k$ ggf. in $x_j = 0$ nicht vermeiden.) Wiederhole das, bis keine Variablen mehr als Summanden vorkommen.

(b) Ersetze Summanden uax, $u \in \Sigma^+$, mit einer neuen Variablen y durch uy und die neue Gleichung $y = ax$. Wiederhole das, bis in allen Summanden ux der Faktor ein $u \in \Sigma$ ist.

(c) Ersetze Summanden au, $u \in \Sigma^*$, mit einer neuen Variablen y durch ay und eine neue Gleichung $y = u$. Wiederhole das, bis 1 der einzige Summand ohne variablen Faktor ist.

Die Schritte (b) und (c) verändern die Menge der Lösungen in den Variablen des Ausgangssystems nicht. In Schritt (a) ist jede Lösung des neuen Systems eine Lösung des ursprünglichen Systems, denn wegen der Idempotenz der Addition folgt $x = r + x$ aus $x = r + 0$. Da die Lösungsmengen bei diesem Schritt verkleinert werden, stimmen die kleinsten Lösungen der beiden Systeme (in den gemeinsamen Variablen) überein. Das Verfahren endet nach endlich vielen Schritten, da man in den nachfolgenden Schritten das vorher Erreichte nicht zerstört. $\square$

Beispiel 4.9 Wir transformieren das rechtslineare Gleichungssystem

$$\begin{array}{rcll} x_0 & = & abx_0 + x_1 + bx_2 + x_3 & =: r_0 \\ x_1 & = & x_0 + x_1 + bbx_1 + ax_2 + bx_2 & =: r_1 \\ x_2 & = & bx_0 + aax_1 + x_3 + 1 & =: r_2 \\ x_3 & = & ax_1 + x_2 + bcx_3 + cd & =: r_3 \end{array}$$

über $\Sigma = \{a, b, c, d\}$ nach dem Beweis von 4.8. (a) Zuerst werden alle Summanden beseitigt, die nur aus einer Variablen bestehen. Die Relation $\rightarrow$ enthält die Zyklen $x_0 \rightarrow x_1 \rightarrow x_1 \rightarrow x_0$ und $x_2 \rightarrow x_3 \rightarrow x_2$. Angenommen, man will vom ersten Zyklus nur die Variable x_0 beibehalten. Man bildet r_1', indem man in r_1 die Summanden x_0, x_1 durch 0 ersetzt:

$$r_1' \;=\; bbx_1 + ax_2 + bx_2$$

Nun wird die Gleichung für x_1 weggelassen, Summanden x_1 in den übrigen Gleichungen durch r_1' ersetzt, und dann die Faktoren x_1 durch Faktoren x_0 ersetzt:

$$\begin{array}{rcl} x_0 & = & abx_0 + (bbx_0 + ax_2 + bx_2) + bx_2 + x_3 \\ x_2 & = & bx_0 + aax_0 + x_3 + 1 \\ x_3 & = & ax_0 + x_2 + bcx_3 + cd \end{array}$$

Entsprechend behandeln wir den zweiten Zyklus $x_2 \rightarrow x_3 \rightarrow x_2$, indem wir die Gleichung für x_3 weglassen und in den übrigen Gleichungen die Summanden x_3 durch $(ax_0 + 0 + bcx_3 + cd)$ und Faktoren x_3 durch Faktoren x_2 ersetzen:

$$\begin{array}{rcl} x_0 & = & abx_0 + (bbx_0 + ax_2 + bx_2) + bx_2 + (ax_0 + bcx_2 + cd) \\ x_2 & = & bx_0 + aax_0 + (ax_0 + bcx_2 + cd) + 1 \end{array}$$

Jetzt treten keine Variablen mehr als Summanden auf. (b) Nun werden die Summanden ux_i mit Faktoren $u \in \Sigma^+$ einer Länge > 1 beseitigt, d.h. abx_0, bbx_0, bcx_2 und aax_0. Man erhält

$$\begin{array}{rcl} x_0 & = & ax_5 + bx_5 + ax_2 + bx_2 + ax_0 + bx_6 + cd \\ x_2 & = & bx_0 + ax_7 + ax_0 + bx_6 + cd + 1 \\ x_5 & = & bx_0 \end{array} \qquad \begin{array}{rcl} x_6 & = & cx_2 \\ x_7 & = & ax_0 \end{array}$$

(c) Zuletzt beiseitigen wir die Summanden $u \in \Sigma^+$, hier also cd, und erhalten

$$\begin{array}{rcl} x_0 & = & ax_5 + bx_5 + ax_2 + bx_2 + ax_0 + bx_6 + cx_8 \\ x_2 & = & bx_0 + ax_7 + ax_0 + bx_6 + cx_8 + 1 \\ x_5 & = & bx_0 \\ x_6 & = & cx_2 \end{array} \qquad \begin{array}{rcl} x_7 & = & ax_0 \\ x_8 & = & dx_9 \\ x_9 & = & 1. \end{array}$$

Dieses Gleichungssystem hat die Form, die bei der Darstellung von Automaten durch Gleichungssysteme wie im Beweis von 4.5 entsteht.

Einen Summanden x_j in einer Gleichung $x_i = r_i(x_1, \ldots, x_m)$ kann man in einem Automaten als einen neuartigen SPONTANEN Übergang vom Zustand x_i in den Zustand x_j

interpretieren, d.h. einen Übergang, der von der Eingabe unabhängig ist. Endliche Automaten mit spontanen Übergängen schreiben wir als

$$\mathcal{A} = (Q, \vdash_a, \vdash_\varepsilon, I, F)_{a \in \Sigma},$$

wobei $\vdash_\varepsilon$ die Relation der spontanen Übergänge ist. Beachte, daß $\vdash_\varepsilon$ im allgemeinen *nicht* die Identität 1_Q auf Q ist. Wenn man Automaten als Relationenalgebren behandelt, kollidiert diese Bezeichnung mit der früheren Konvention $\vdash_w = 1_Q$ im Fall $w \equiv \varepsilon$.

Rechtslineare Gleichungssysteme kann man als (rechtlineare) *Grammatiken* auffassen: die Variablen stehen für die definierten Ausdrucksmengen, d.h. die syntaktischen Kategorien einer Grammatik, und die Summanden einer Gleichung

$$x_i = r_j = \ldots ... + ux_j + v + \ldots$$

für Konstruktionsmöglichkeiten von Ausdrücken der Kategorie x_i, die man als getrennte (Ersetzungs-)*Regeln*

$$x_i \to ux_j \qquad \text{und} \qquad x_i \to v$$

schreibt. Sie entsprechen dem Anfügen eines Präfixes $u \in \Sigma^*$ an einen Ausdruck einer Kategorie x_j und der expliziten Angabe eines Ausdrucks $v \in \Sigma^*$. Zur Definition von komplexeren, z.B. natürlichen Sprachen, braucht man weitere Konstruktionsmöglichkeiten.

Definition 4.10 Eine RECHTSLINEARE GRAMMATIK $G = (V, \Sigma, S, P)$ besteht aus disjunkten endlichen Mengen V und Σ, den VARIABLEN oder NONTERMINALSYMBOLEN und den KONSTANTEN oder TERMINALSYMBOLEN, einem STARTSYMBOL $S \in V$, und einer endlichen Menge $P \subseteq V \times (\Sigma^* V \cup \Sigma^*)$ von REGELN. Eine Regel (x, uy) bzw. (x, u) schreibt man $x \to uy$ bzw. $x \to u$. Die VON G ERZEUGTE SPRACHE ist

$$L(G) := \{\, w \in \Sigma^* \mid S \Rightarrow_P^n w \text{ für ein } n \in \mathbb{N} \,\},$$

wobei $ux \Rightarrow_P^0 w :\Longleftrightarrow ux = w$ und $ux \Rightarrow_P^{n+1} w :\Longleftrightarrow$ es gibt $(vy, w) \in P$ mit $ux \Rightarrow_P^n vy$.

Beispiel 4.11 Das rechtslineare Gleichungssystem aus Beispiel 4.9 hat als rechtslineare Grammatik dargestellt die folgende Regelmenge:

$$\begin{array}{lll@{\qquad}lll@{\qquad}lll@{\qquad}lll}
x_0 & \to & abx_0 & x_1 & \to & x_0 & x_2 & \to & bx_0 & x_3 & \to & ax_1 \\
x_0 & \to & x_1 & x_1 & \to & x_1 & x_2 & \to & aax_1 & x_3 & \to & x_2 \\
x_0 & \to & bx_2 & x_1 & \to & bbx_1 & x_2 & \to & x_3 & x_3 & \to & bcx_3 \\
x_0 & \to & x_3 & x_1 & \to & ax_2 & x_2 & \to & \epsilon & x_3 & \to & cd \\
 & & & x_1 & \to & bx_2 & & & & & &
\end{array}$$

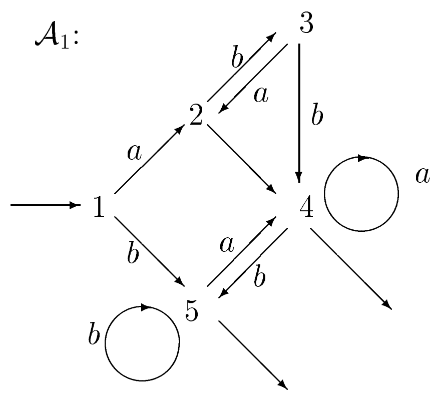
Summanden 1 in den Gleichungen werden in den Regeln als ϵ geschrieben.

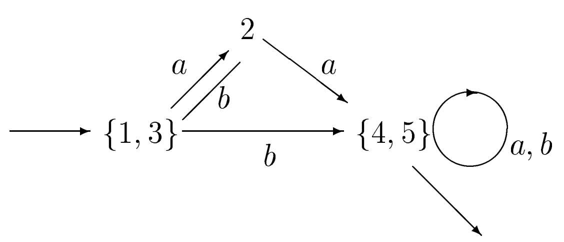
4.3 Endliche Automaten mit minimaler Zustandsanzahl

In der Praxis ist es wegen Speicherplatz und Laufzeitverhalten oft wichtig, daß man einen möglichst kleinen deterministischen Automaten verwendet.

Wir zeigen jetzt, daß es zu jedem deterministischen endlichen Automaten $\mathcal{A}$ einen deterministischen Automaten $\mathcal{A}'$ mit $L(\mathcal{A}) = L(\mathcal{A}')$ und möglichst kleiner Zustandszahl gibt. Er kann aus $\mathcal{A}$ berechnet werden und ist bis auf Graphenisomorphie durch $L(\mathcal{A})$ eindeutig bestimmt.

Beispiel 4.12 Folgende Automaten $\mathcal{A}_1$ und $\mathcal{A}_2$ erkennen dieselbe Sprache:





An den Gleichungssystemen der Automaten,

$$\begin{array}{rcl} x_1 &=& ax_2 + bx_5 \\ x_2 &=& ax_4 + bx_3 \\ x_3 &=& ax_2 + bx_4 \\ x_4 &=& 1 + ax_4 + bx_5 \\ x_5 &=& 1 + ax_4 + bx_5 \end{array} \qquad \text{und} \qquad \begin{array}{rcl} x_{1,3} &=& ax_2 + bx_{4,5} \\ x_2 &=& ax_{4,5} + bx_{1,3} \\ x_{4,5} &=& 1 + ax_{4,5} + bx_{4,5}, \end{array}$$

erkennt man nämlich, daß für jede Lösung des linken Systems zu $\mathcal{A}_1$ zunächst $x_5 = x_4$ und damit auch $x_3 = x_1$ ist, so daß man das rechte Gleichungssystem für a_2 bekommt.

Die Sprache des Automaten darf nicht verändert werden, wenn wir Zustände p, q miteinander identifizieren. Wir identifizieren alle Zustände, die dieselbe Sprache definieren:

$$p \sim^{\mathcal{A}} q \quad :\Longleftrightarrow \quad \forall w \in \Sigma^* \ (\delta(p, w) \in F \leftrightarrow \delta(q, w) \in F).$$

Beachte, daß

$$p \sim^{\mathcal{A}} q \quad \Longrightarrow \quad \delta(p, a) \sim^{\mathcal{A}} \delta(q, a) \text{ für jedes } a \in \Sigma,$$

da für jedes $w = aw'$

$$\delta(\delta(p, a), w') = \delta(p, w) \in F \quad \Longleftrightarrow \quad \delta(\delta(q, a), w') = \delta(q, w) \in F.$$

Daher können wir δ von $\mathcal{A}$ leicht auf die identifizierten Zustände übertragen. Damit der neue Automat nur von $L(\mathcal{A})$, aber nicht von $\mathcal{A}$ abhängt, betrachten wir statt $\sim^{\mathcal{A}}$ die folgende Äquivalenzrelation $\sim_L \subseteq \Sigma^* \times \Sigma^*$ für $L = L(\mathcal{A})$, die analog zu $\sim^{\mathcal{A}}$ definiert ist:

Definition 4.13 Definiere zu $L \subseteq \Sigma^*$ eine Äquivalenzrelation $\sim_L$ auf Σ^* durch

$$v_1 \sim_L v_2 \quad \Longleftrightarrow \quad \forall w \in \Sigma^* \, (v_1 w \in L \iff v_2 w \in L).$$

Beachte, daß $\sim_L$ RECHTSINVARIANT ist, d.h. für alle $v_1, v_2 \in \Sigma^*$ ist

$$v_1 \sim_L v_2 \quad \Longrightarrow \quad v_1 w \sim_L v_2 w \text{ für alle } w \in \Sigma^*.$$

Satz 4.14 (Nerode 1958) L wird genau dann von einem deterministischen endlichen Automaten erkannt, wenn der INDEX VON $\sim_L$, d.h. die Anzahl der $\sim_L$-Äquivalenzklassen, endlich ist.

Beweis: $\Longrightarrow$: Sei $L = L(\mathcal{A})$ für den deterministischen Automaten $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$. Definiere eine Äquivalenzrelation $\sim_\mathcal{A}$ auf Σ^* durch $v_1 \sim_\mathcal{A} v_2 \iff \delta(q_0, v_1) = \delta(q_0, v_2)$. Für alle $v_1, v_2 \in \Sigma^*$ gilt dann:

$$\begin{aligned} v_1 \sim_\mathcal{A} v_2 \quad &\Longrightarrow \quad \forall w \in \Sigma^* \, (\delta(q_0, v_1 w) = \delta(\delta(q_0, v_i), w) = \delta(q_0, v_2 w)) \\ &\Longrightarrow \quad \forall w \in \Sigma^* (v_1 w \in L(\mathcal{A}) \iff v_2 w \in L(\mathcal{A})) \\ &\Longrightarrow \quad v_1 \sim_L v_2. \end{aligned}$$

Wenn also $[v_1]_{\sim_L} \neq [v_2]_{\sim_L}$ ist, so ist $[v_1]_{\sim_\mathcal{A}} \neq [v_2]_{\sim_\mathcal{A}}$, und daher

$$\text{Index von } \sim_L \leq \text{ Index von } \sim_\mathcal{A} \leq |Q^\mathcal{A}| < \infty.$$

$\Longleftarrow$: Sei n der Index von L, und $w_1, w_2, \ldots, w_n$ seien paarweise nicht äquivalent bzgl. $\sim_L$, wobei wir $w_1 = \varepsilon$ annehmen können. Sei $[w] := \{ v \in \Sigma^* \mid w \sim_L v \}$ die Äquivalenzklasse von w bezüglich $\sim_L$. Definiere $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ durch

$$\begin{array}{lll} Q := \{[w_1], \ldots, [w_n]\}, & F & := \{ [w_i] \mid i \in \{1, \ldots, n\}, \, w_i \in L \} \\ q_0 := [\varepsilon], & \delta([w_i], a) & := [w_i a], \quad \text{für } a \in \Sigma. \end{array}$$

Da $\sim_L$ rechtsinvariant ist, ist δ wohldefiniert, d.h. unabhängig von der Wahl der w_i:

$$[v_1] = [v_2] \quad \Longrightarrow \quad v_1 \sim_L v_2 \quad \Longrightarrow \quad v_1 a \sim_L v_2 a \quad \Longrightarrow \quad [v_1 a] = [v_2 a].$$

Offenbar ist $\mathcal{A}$ ein deterministischer endlicher Automat. Wegen $\delta(q_0, w) = [w]$ und

$$\begin{aligned} w \in L(\mathcal{A}) \quad &\iff \quad \delta(q_0, w) = [w] \in F \\ &\iff \quad \bigvee_{i=1}^{n} (w \sim_L w_i \wedge w_i \in L) \\ &\iff \quad \exists v \in L \; w \sim_L v \quad \iff \quad w \in L \end{aligned}$$

erkennt $\mathcal{A}$ die Sprache L. □

Korollar 4.15 Sei $L = L(\mathcal{A})$ für einen deterministischen endlichen Automaten $\mathcal{A}$ und $\mathcal{B}$ der im vorigen Beweis konstruierte Automat aus $\sim_L$-Äquivalenzklassen.

(i) $\mathcal{A}$ hat mindestens so viele Zustände wie $\mathcal{B}$.

(ii) Falls $\mathcal{A}$ genauso viele Zustände wie $\mathcal{B}$ hat, so ist $\mathcal{A}$ zu $\mathcal{B}$ isomorph.

Beweis: (i) $|Q^{\mathcal{B}}| =$ Index von $\sim_L \leq$ Index von $\sim_{\mathcal{A}} \leq |Q^{\mathcal{A}}|$.

(ii) Zu jedem $q \in Q^{\mathcal{A}}$ gibt es ein $v_q \in \Sigma^*$ mit $\delta(q_0, v_q) = q$, da man den Zustand q sonst weglassen könnte. Ordne jedem Zustand $q \in Q^{\mathcal{A}}$ den Zustand $[v_q]$ des Automaten $\mathcal{B}$ zu. Dann ist

$$\delta^{\mathcal{A}}(q_0, v_{\delta^{\mathcal{A}}(q,a)}) = \delta^{\mathcal{A}}(q, a) = \delta^{\mathcal{A}}(\delta^{\mathcal{A}}(q_0, v_q), a) = \delta^{\mathcal{A}}(q_0, v_q a),$$

also $v_q a \sim_L v_{\delta^{\mathcal{A}}(q,a)}$ und daher $\delta^{\mathcal{B}}([v_q], a) = [v_q a] = [v_{\delta^{\mathcal{A}}(q,a)}]$. Jedem Zustand und jedem Übergang in $\mathcal{B}$ entspricht also ein Zustand bzw. Übergang in $\mathcal{B}$ und umgekehrt. Man überlegt sich leicht, daß auch die Endzustandsmengen einander entsprechen. $\square$

Nach 4.15 hat der aus den $\sim_L$-Äquivalenzklassen konstruierte Automat eine minimale Anzahl von Zuständen. Wir wollen jetzt noch sehen, wie man diesen Automaten aus einem schon bekannten Automaten $\mathcal{A}$ mit $L(\mathcal{A}) = L$ konstruieren kann.

Satz 4.16 Aus jedem deterministischen endlichen Automaten $\mathcal{A}$ kann man einen äquivalenten, zustandsminimalen deterministischen Automaten $\mathcal{A}'$ konstruieren.

Beweis: Sei $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$. Definiere Äquivalenzrelationen $\sim_n$ auf Q durch

$$\begin{array}{lll} p \sim_o q & :\iff & p \in F \leftrightarrow q \in F \\ p \sim_{n+1} q & :\iff & p \sim_n q \ \wedge\ \forall a \in \Sigma.\ \delta(p, a) \sim_n \delta(q, a). \end{array}$$

Sei $\Sigma^{\leq n} := \{\, v \in \Sigma^* \mid \text{ Die Länge von } v \text{ ist } \leq n \,\}$. Dann gilt

$$p \sim_n q \quad \iff \quad \forall w \in \Sigma^{\leq n}(\delta(p, w) \in F \leftrightarrow \delta(q, w) \in F).$$

Da Q endlich ist und $\sim_{n+1} \subseteq \sim_n$, gibt es unter diesen Äquivalenzrelationen eine feinste, d.h. ein n mit $\sim_n = \sim_{n+k}$ für alle k. Für diese Relation $\sim$ gilt also

$$p \sim q \quad \iff \quad \forall w \in \Sigma^*(\delta(p, w) \in F \leftrightarrow \delta(q, w) \in F).$$

Wähle $\mathcal{A}' := (Q', \Sigma, q_0, \delta', F')$ wie folgt, mit $[q] := \{\, p \in Q \mid p \sim q \,\}$:

$$\begin{array}{lll} Q' & := & \{\, [q] \mid q \in Q,\ \exists v \in \Sigma^*\ \delta(q_0, v) = q \,\}, \\ q_0' & := & [q_0] \end{array} \qquad \begin{array}{lll} F' & := & \{\, [q] \in Q' \mid q \in F \,\} \\ \delta'([q], a) & := & [\delta(q, a)] \end{array}$$

Beachte, daß für $p_i = \delta(q_0, v_i)$ jetzt gilt:

$$\begin{aligned} p_1 \sim p_2 \quad &\Longleftrightarrow \quad \forall w \in \Sigma^*(\delta(q_0, v_1 w) \in F \leftrightarrow \delta(q_0, v_2 w) \in F) \\ &\Longleftrightarrow \quad v_1 \sim_{L(\mathcal{A})} v_2. \end{aligned}$$

Da wegen $\sim\, = \,\sim_n\, = \,\sim_{n+k}$ für alle k auch

$$p_1 \sim p_2 \quad \Longrightarrow \quad \forall a \in \Sigma.\ \delta(p_1, a) \sim \delta(p_2, a)$$

gilt, ist δ' wohldefiniert, und offenbar

$$\delta'([p_i], a) = [\delta(p_i, a)] = [v_i a]_{\sim_{L(\mathcal{A})}}.$$

Damit ist $\mathcal{A}'$ isomorph zu dem im Satz von Nerode aus den Äquivalenzklassen von $\sim_L$ definierten Automaten. Also ist $L(\mathcal{A}') = L(\mathcal{A})$, und es gibt keinen deterministischen endlichen Automaten für $L(\mathcal{A})$, der weniger Zustände als $\mathcal{A}'$ hat. □

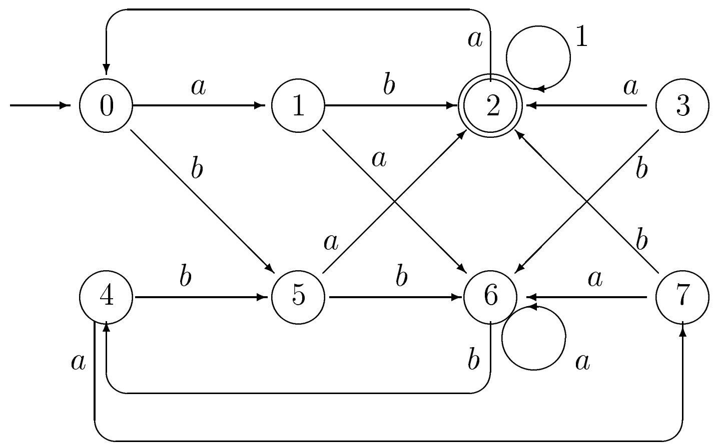
Beispiel 4.17 Wir berechnen die Äquivalenzrelation $\sim_n$, bis $\sim_{n+1} = \sim_n$, gemäß der obigen induktiven Definition, und bilden dann mit $\sim_n = \sim$ den Automaten $\mathcal{A}'$ wie angegeben. Zur Berechnung von $\sim_{n+1}$ aus $\sim_n$ betrachte jede Äquivalenzklasse B von $\sim_n$ und zerlege B in Unterklassen $B_1, B_2, \ldots$, wenn Elemente von B durch ein $a \in \Sigma$ in verschiedene $\sim_n$-Klassen führen. Dazu stellen wir eine Tabelle

		q_0	$\ldots$	q_j	$\ldots$	q_m
$\sim_0$			F		$-F$	
		$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	a_1			$[\delta(q_j, a_1)]_n$		
$n+1$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	a_k			$[\delta(q_j, a_k)]_n$		
$\sim_{n+1}$				$[q_j]_{n+1}$		
		$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

auf. In der ersten Zeile wird die Zugehörigkeit der Zustände zu den beiden $\sim_0$-Äquivalenzklassen F und $-F := Q - F$ eingetragen. Die Äquivalenzklassen $[q]_{n+1}$ der Relation $\sim_{n+1}$ in Zeile $n+1$ werden durch

$$[q]_{n+1} = \{\, p \in [q]_n \mid [\delta(p, a_i)]_n = [\delta(q, a_i)]_n \text{ für } i = 1, \ldots, k \,\}$$

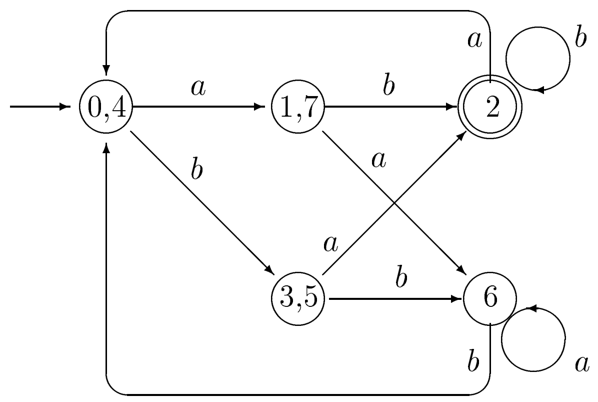
aus der vorigen Zeile berechnet, d.h. Zustände derselben $\sim_n$-Klasse mit gleichen Einträgen in Zeile $n+1$ kommen in dieselbe $\sim_{n+1}$-Klasse. Einelementige Klassen brauchen nicht mehr betrachtet zu werden, da sie sich nicht weiter aufspalten lassen.

$\mathcal{A} =$ 

Aus den $\sim_0$-Klassen: $F = \{2\}$ und $-F = \{0, 1, 3, 4, 5, 6, 7\}$ berechnet man:

		0	1	2	3	4	5	6	7
$\sim_0$		$-F$	$-F$	F	$-F$	$-F$	$-F$	$-F$	$-F$
1	a	$-F$	$-F$		F	$-F$	F	$-F$	$-F$
	b	$-F$	F		$-F$	$-F$	$-F$	$-F$	F
$\sim_1$		A	B	F	D	A	D	A	B
2	a	B	A		F	B	F	A	A
	b	D	F		A	D	A	A	F
$\sim_2$		E	B	F	D	E	D	G	B
3	a	B	G		F	B	F		G
	b	D	F		G	D	G		F
$\sim_3$		E	B	F	D	E	D	G	B

Die $\sim_0$-Klasse $-F$ zerfällt in die drei $\sim_1$-Klassen $A = \{0, 4, 6\}$, $B = \{1, 7\}$ und $D = \{3, 5\}$, wovon A in die $\sim_2$-Klassen $E = \{0, 4\}$ und $G = \{6\}$ zerfällt, während B und D sich nicht weiter aufspalten. Die $\sim_2$-Klassen B, E, D sind auch $\sim_3$-Klassen, so daß man wegen $\delta^{\mathcal{A}'}([q], a_i) = [\delta(q, a_i)]$ den folgenden Automaten erhält:

$\mathcal{A}' =$ 

Sprachen mit nur endlich vielen Distributionsklassen

Eine Variante der Konstruktion von Nerode ist auch vom linguistischen Standpunkt interessant. Als Test, ob ein Ausdruck "von einer bestimmten syntaktischen Kategorie" ist, überprüft man in der Linguistik, ob er in denselben Satzkontexten vorkommt wie die anderen Ausdrücke dieser Kategorie. Nominalphrasen sollten z.B. an den gleichen Positionen in Sätzen auftreten wie die (Pro-) Nomina. Das können wir wie folgt präzisieren.

Definition 4.18 Zwei Wörter w_1 und $w_2 \in \Sigma^*$ haben dieselbe DISTRIBUTION bezüglich der Sprache $L \subseteq \Sigma^*$, in Zeichen: $w_1 \simeq_L w_2$, falls

$$\forall u \in \Sigma^* \forall v \in \Sigma^* (uw_1v \in L \iff uw_2v \in L).$$

Was eine "syntaktische Kategorie" ist, wird normalerweise in einer Grammatik, also etwas willkürlich, festgelegt. Identifiziert man aber syntaktische Kategorien mit den Distributionsklassen, so hat man "objektive", d.h. allein durch die Sprache L festgelegte Kategorien. Die regulären Sprachen sind nun genau diejenigen Sprachen, die nur endlich viele solcher syntaktischen Kategorien haben:

Satz 4.19 (Myhill 1957) Eine Sprache $L \subseteq \Sigma^*$ ist genau dann regulär, wenn die Relation $\simeq_L$ nur endlich viele Äquivalenzklassen hat.

Beweis: $\Longrightarrow$: Sei $L = L(\mathcal{A})$ für einen endlichen Automaten $\mathcal{A}$. Definiere

$$w_1 \simeq_{\mathcal{A}} w_2 \iff \vdash_{w_1} = \vdash_{w_2} .$$

Man rechnet leicht nach, daß für alle $w_1, w_2 \in \Sigma^*$ gilt:

$$w_1 \simeq_{\mathcal{A}} w_2 \Longrightarrow w_1 \simeq_L w_2.$$

Da es nur endlich viele Relationen $\vdash_w \subseteq Q \times Q$ gibt, hat $\simeq_{\mathcal{A}}$ und damit auch $\simeq_L$ nur endlich viele Äquivalenzklassen.

$\Longleftarrow$: Definiere $\mathcal{B}$ ähnlich wie im Satz von Nerode aus den endlich vielen Äquivalenzklassen von $\simeq_L$ statt $\sim_L$. Dies geht, da wegen

$$\vdash_{w_1} = \vdash_{w_2} \quad \Longrightarrow \quad \forall v \in \Sigma^* : \vdash_{w_1 v} = \vdash_{w_1} \circ \vdash_v = \vdash_{w_2} \circ \vdash_v = \vdash_{w_2 v}$$

auch $\simeq_L$ rechtsinvariant ist. □

4.4 Wiederholungseigenschaften regulärer Sprachen

Mit Hilfe der deterministischen Automaten kann man leicht zeigen, daß lange Wörter in regulären Sprachen gewisse Teilwörter haben, deren Wiederholung wieder zu Wörtern der Sprache führt.

Lemma 4.20 (Wiederholungslemma für reguläre Sprachen) Zu jeder regulären Sprache $L \subseteq \Sigma^*$ gibt es eine Zahl n, so daß zu jedem Wort $x \in L$ mit $|x| > n$ geeignete $u, v, w \in \Sigma^*$ existieren mit

i) $x = uvw$, ii) $n \geq |v| > 0$ und iii) $uv^k w \in L$ für alle $k \geq 0$.

Beweis: Sei $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein deterministischer endlicher Automat mit $L = L(\mathcal{A})$ und $n := |Q|$. Sei $x = a_1 \dots a_m$ mit $m > n$. Die Zustände $\delta(q_0, a_1), \dots, \delta(q_0, a_1 \dots a_m)$ können wegen $m > |Q|$ nicht paarweise verschieden sein. Also gibt es $1 \leq i < j \leq m$ mit $\delta(q_0, a_1 \dots a_i) = \delta(q_0, a_1 \dots a_i\, a_{i+1} \dots a_j)$. Sei $j > i$ minimal mit dieser Eigenschaft. Für $u = a_0 \dots a_i$, $v = a_{i+1} \dots a_j$ und $w = a_{j+1} \dots a_m$ gilt dann die Behauptung, da $\delta(q_0, u) = \delta(\delta(q_0, u), v)$. □

Eine Sprache, die diese Wiederholungseigenschaft nicht hat, kann also nicht regulär sein. Damit kann man die Nichtregularität gewisser Sprachen nachweisen.

Korollar 4.21 Die Sprache $L := \{\, a^n\, b^n \mid n \in I\!N \,\} \subseteq \{a, b\}^*$ ist nicht regulär.

Beweis: Angenommen, L sei regulär, und für ein hinreichend großes Wort $a^n b^n$ habe man eine Zerlegung $a^n b^n = uvw$ wie im Lemma. Wäre $v \in L(aa^*)$, so enthielte $uvvw$ mehr Vorkommen von a als von b und wäre damit nicht in L. Analog schließt man den Fall $v \in L(bb^*)$ aus. Würde v sowohl Vorkommen von a als auch von b haben, so würde aber $uvvw$ das Teilwort ba enthalten, und wäre damit ebenfalls nicht in L. □

Korollar 4.22 $L(\mathcal{A})$ ist genau dann unendlich, wenn $\exists w \in L(\mathcal{A}) \quad |Q| < |w| \leq 2 \cdot |Q|$.

Beweis: Sei n die Anzahl der Zustände von $\mathcal{A}$. Falls $L(\mathcal{A})$ unendlich ist, gibt es $x \in L(\mathcal{A})$ mit $|x| > 2n$. Nach Lemma 4.20 gibt es eine Zerlegung $x = uvw$, so daß $uw \in L(\mathcal{A})$ und $|x| - n \leq |uw| < |x|$. Durch Wiederholung erreichen wir so ein Wort in $L(\mathcal{A})$ mit der behaupteten Länge. Ist umgekehrt $x \in L(\mathcal{A})$ mit $|Q| < |w| \leq 2|Q|$, so ist nach 4.20 für eine geeignete Zerlegung $x = uvw$ mit $v \neq \varepsilon$ auch $\{\, uv^k w \mid k \in I\!N \,\} \subseteq L(\mathcal{A})$ und damit $L(\mathcal{A})$ unendlich. □

4.5 Abschluß- und Entscheidbarkeitseigenschaften

Aus der Äquivalenz zwischen regulären und den von endlichen Automaten erkannten Sprachen erhalten wir einige Abschlußeigenschaften der Klasse der regulären Sprachen über Σ.

Satz 4.23 Die Klasse Reg_Σ der regulären Sprachen über Σ ist abgeschlossen unter den mengentheoretischen Operationen $\cup$, $\cap$ und $-$, den regulären Operationen $\cdot$ und *, sowie den Divisionen $\backslash$ und $/$.

Beweis: Seien L, L_1 und L_2 reguläre Sprachen über Σ. Ist $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein deterministischer endlicher Automat mit $L = L(\mathcal{A})$, so wird $\overline{L} := \Sigma^* - L$ vom endlichen Automaten $(Q, \Sigma, \delta, q_0, Q - F)$ erkannt, ist also regulär.

Mit der Komplementbildung sind aber auch Durchschnitt und Differenz regulärer Sprachen regulär, da $L_1 \cap L_2 = \overline{(\overline{L_1} \cup \overline{L_2})}$ und $L_1 - L_2 = L_1 \cap \overline{L_2}$.

Der Quotient L_1/L_2 ist regulär, falls L_1 regulär ist, wobei $L_2 \subseteq \Sigma^*$ nicht regulär sein muß. Sei $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein deterministischer endlicher Automat, der L_1 erkennt. Der Automat

$$\mathcal{A}' = (Q, \Sigma, \delta, q_0, F') \quad \text{mit} \quad F' := \{\, q \in Q \mid \forall v \in L_2\ \delta(q, v) \in F \,\}$$

erkennt L_1/L_2, denn:

$$\begin{array}{rclcl} w \in L_1/L_2 & \iff & \forall v \in L_2\ wv \in L_1 & \iff & \forall v \in L_2\ \delta(q_0, wv) \in F \\ & \iff & \forall v \in L_2\ \delta(\delta(q_0, w), v) \in F & \iff & \delta(q_0, w) \in F' \\ & \iff & w \in L(\mathcal{A}'). & & \end{array}$$

Für die Sprache $L_2 \backslash L_1$ machen wir einen Umweg über die Komplemente und benutzen

$$w \notin L_2 \backslash L_1 \iff \neg \forall v \in L_2\ vw \in L_1 \iff \exists v \in L_2\ vw \notin L_1.$$

Sei $\mathcal{A} = (Q, \vdash_a, q_0, F)_{a \in \Sigma}$ ein endlicher Automat, der das Komplement von L_1 erkennt. Mit einem neuen Anfangszustand $q_0' \notin Q$ konstruieren wir einen endlichen Automaten $\mathcal{A}'' = (Q \cup \{q_0'\}, \vdash_a, \vdash_\varepsilon, q_0', F)_{a \in \Sigma}$ mit spontanen Übergängen, indem wir für $q \in Q$

$$q_0' \vdash_\varepsilon q \iff \exists v \in L_2\ q_0 \vdash_v q$$

setzen. $\mathcal{A}''$ kann also von q_0' spontan in einen beliebigen Zustand von Q übergehen, den wir mit $\mathcal{A}$ durch Lesen eines $v \in L_2$ erreichen können. Dann ist

$$\begin{array}{rcl} w \notin L_2 \backslash L_1 & \iff & \exists v \in L_2\ vw \notin L_1 \\ & \iff & \exists v \in L_2\, \exists q, q_e \in Q\ q_0 \vdash_v q \vdash_w q_e \in F \\ & \iff & \exists q, q_e \in Q\ q_0' \vdash_\varepsilon q \vdash_w q_e \in F \\ & \iff & w \text{ wird von } \mathcal{A}'' \text{ erkannt.} \end{array}$$

Also ist $\Sigma^* - L_2 \backslash L_1 = L(\mathcal{A}'')$ und damit auch sein Komplement $L_2 \backslash L_1$ regulär. Beachte aber, daß wir F' für $\mathcal{A}'$ und $\vdash_\varepsilon$ für $\mathcal{A}''$ nicht effektiv angeben können. $\square$

Proposition 4.24 Die Klasse der regulären Sprachen über Σ ist abgeschlossen unter Substitution, d.h. Ersetzung von Buchstaben durch reguläre Sprachen über Σ.

Beweis: Sei $\Sigma = \{a_1, \ldots, a_n\}$ und $L, L_1, \ldots, L_n \subseteq \Sigma^*$ durch die regulären Ausdrücke $r, r_1, \ldots, r_n$ definiert. Dann ist

$$L[L_1, \ldots, L_n] := \bigcup \{ L_{i_1} \cdots L_{i_k} \mid a_{i_1} \cdots a_{i_k} \in L, 1 \leq i_1, \ldots, i_k \leq n \}$$

durch den regulären Ausdruck $r[r_1/a_1, \ldots, r_n/a_n]$ definiert, wie man durch Induktion über den Aufbau von r sieht. □

Aus der Konstruktion minimaler deterministischer endlicher Automaten erhalten wir ein Entscheidungsverfahren für die Sprachgleichheit regulärer Ausdrücke:

Satz 4.25 Es gibt Verfahren, mit denen man für reguläre Ausdrücke r und s über Σ und $w \in \Sigma^*$ feststellen kann, welche der folgenden Aussagen gelten:

i) $L(r) = L(s)$,	ii) $L(r) = \emptyset$,	iii) $L(r) = \Sigma^*$,
iv) $L(r) \subseteq L(s)$,	v) $w \in L(r)$,	vi) $\mid L(r) \mid < \infty$.

Beweis: Nach 4.22 können wir vi) feststellen. Die Behauptungen für ii) – v) folgen aus i), wenn man die Gleichungen $r = 0$, $r = (a_1 + \ldots + a_n)^*$, $r + s = s$ und $w + r = r$ benutzt.

Bleibt i) zu zeigen. Man konstruiere aus r nach Satz 4.5 einen endlichen Automaten $\mathcal{A}_r$ mit $L(r) = L(\mathcal{A}_r)$. Wir können zu $\mathcal{A}_r$ einen deterministischen endlichen Automaten $\mathcal{B}_r$ mit $L(\mathcal{A}_r) = L(\mathcal{B}_r)$ und minimal vielen Zuständen angeben. Sei $\mathcal{B}_s$ der entsprechend konstruierte Automat zu s. Da ein deterministischer endlicher Automat minimaler Zustandsanzahl bis auf Isomorphie eindeutig durch die erkannte Sprache bestimmt ist, ist $L(r) = L(s) \iff \mathcal{B}_r \simeq \mathcal{B}_s$. Die (Nicht-) Isomorphie zweier Automaten kann man aber an Hand der Übergangstabellen feststellen. □

Der Satz gilt natürlich auch, wenn wir statt r oder s einen endlichen Automaten vorgeben. Der Aufwand, den man zur Bestimmung der Antwort treiben muß, hängt aber davon ab, ob die jeweilige Frage mit regulären Ausdrücken oder endlichen Automaten gestellt wird.

4.6 Zur Syntaxanalyse regulärer Sprachen

Normalerweise genügt es nicht, nur zu entscheiden, ob Ausdrücke $w \in \Sigma^*$ zu einer Sprache L gehören, sondern man muß auch die grammatische oder syntaktische Struktur der Ausdrücke kennen. Was eine syntaktische Struktur ist, hängt von einer Grammatik, d.h. einer endlichen Darstellung von L, ab.

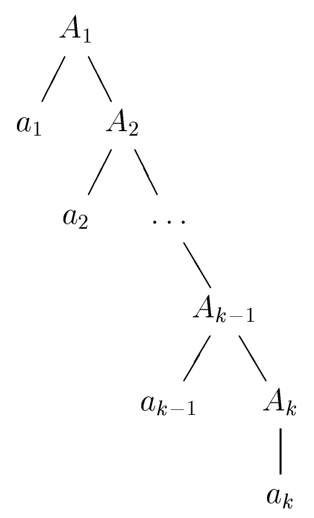
Eine SYNTAKTISCHE STRUKTUR eines Ausdrucks $w \in L$ bezüglich einer Grammatik G von L ist, allgemein gesagt, ein Beweis dafür, daß w zu $L = L(G)$ gehört. Da man von gewissen Beweisdetails, z.B. der Reihenfolge, in der man nachweist, daß Teilausdrücke zu bestimmten Teilsprachen gehören, absehen möchte, ist eine syntaktische Struktur genauer gesagt eine Äquivalenzklasse von Beweisen – wie man die Äquivalenz wählt, hängt natürlich auch von linguistischen Vorstellungen ab. Ein solcher Beweis kann auch eine akzeptierende Berechung durch einen Automaten $\mathcal{A}$ sein.

Für reguläre Sprachen haben wir zwei unterschiedliche Darstellungsweisen betrachtet, die regulären Ausdrücke und die endlichen Automaten bzw. rechtslinearen Grammatiken.

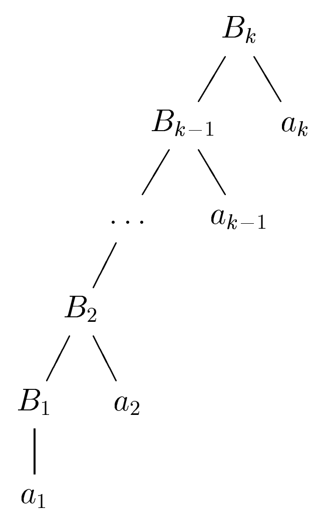
Endliche Automaten oder rechtslineare (bzw. linkslineare) Grammatiken führen zu sehr einfachen, aus linguistischer Sicht unbefriedigenden syntaktischen Strukturen für reguläre Sprachen. Kommt in der Gleichungsdarstellung des Automaten eine Gleichung der Form

$$A_j = \cdots + a_j A_{j+1} + \cdots$$

vor, so ist die Verwendung dieser Konstruktionsmöglichkeit (einen Ausdruck der Kategorie A_{j+1} mit dem Präfix a_j zu versehen) ein erlaubter Beweisschritt. Der Nachweis von $w \in L(\mathcal{A})$ besteht also in der Angabe einer Folge erlaubter Konstruktionsschritte – der akzeptierenden Berechung durch den Automaten. Man kann das graphisch in der einfache Form eines Kamms darstellen:



bzw.



Reguläre Ausdrücke führen zu linguistisch naheliegenderen und besser gegliederten syntaktischen Strukturen für reguläre Sprachen. Ein Wort w gehört zu der durch r definierten Sprache, wenn es sich gemäß dem Aufbau von r in Teilworte zerlegen läßt, die zu den durch entsprechende Teilausdrücke von r definierten Sprachen gehören.

Verschiedene reguläre Ausdrücke r und s über Σ können dieselbe Sprache $L \subseteq \Sigma^*$ definieren; sie liefern dann normalerweise aber verschiedene syntaktische Strukturen:

Sei $r := (ab)^* \cdot a$ und $s := a \cdot (ba)^*$. Jedes Wort $w \in L := L(r) = L(s)$ hat (mindestens) zwei syntaktische Analysen, hier für das Wort $w = aba$:

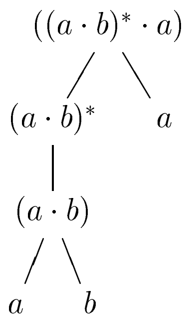
(i) aba wird analysiert als $u \cdot v$ mit $u = (a \cdot b) \in L((ab)^*)$ und $v = a \in L(a)$, und

(ii) aba wird analysiert als $u \cdot v$ mit $u = a \in L(a)$ und $v = (b \cdot a) \in L((ba)^*)$.

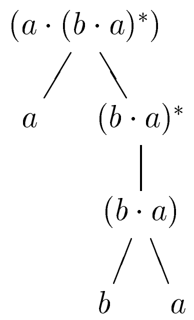
Die Analyse i) entspricht dem Aufbau der Sprache gemäß dem regulären Ausdruck r, während ii) eher zum Aufbau der Sprache gemäß s paßt.

Dem Wort aba werden syntaktische Strukturen zugeordnet, an denen man erkennt, weshalb es zu der durch r bzw. durch s beschriebenen Sprache gehört.

Für i)



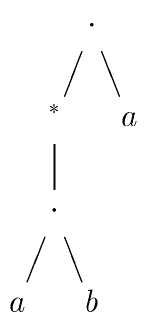
und für ii)



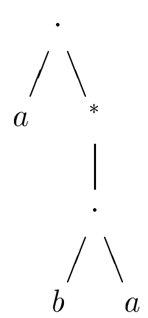
Im vorliegenden Fall sind diese syntaktischen Strukturen also Bäume, die einen Aufbau des Worts w durch bloße *Verkettung* von *zusammenhängenden* Teilstücken beschreiben. (Es ist wesentlich, daß die Zweige des Baumes geordnet sind, damit die Reihenfolge der Argumente der Verkettung richtig erfaßt wird.) Dabei gehören die Teilstücke zu Sprachen, die von Teilausdrücken der Sprachbeschreibungen r bzw. s definiert werden.

Bei der Angabe der syntaktischen Strukturen kann man sich darauf beschränken, die regulären Operationen anzugeben, die ihren Aufbau festlegen, also

für i)



und für ii)



Jeder reguläre Ausdruck r über Σ kann auch als eine Festlegung einer Menge $S(r)$ gewisser syntaktischer Strukturen verstanden werden.

$$S(0) = \emptyset \quad \text{(leere Menge syntaktischer Strukturen)}$$

$$S(a) = \{\cdot a\}$$

$$S(1) = \left\{ \begin{array}{c} 1 \\ | \\ \varepsilon \end{array} \right\}$$

$$S(r \cdot s) = \left\{ \begin{array}{ccc} & (r \cdot s) & \\ & / \quad \backslash & \\ u & & v \end{array} \;\middle|\; u \in S(r),\ v \in S(s) \right\}$$

$$S(r + s) = \left\{ \begin{array}{c} (r + s) \\ | \\ u \end{array} \;\middle|\; u \in S(r) \cup S(s) \right\}$$

$$S(r^*) = \left\{ \begin{array}{ccc} & r^* & \\ / & & \backslash \\ u_1 & \ldots & u_n \end{array} \;\middle|\; n \geq 0,\ u_1 \in S(r), \ldots,\ u_n \in S(r) \right\}$$

Für $n = 0$ wird die n-fache Verzweigung durch $\begin{array}{c} r^* \\ | \\ \varepsilon \end{array}$ dargestellt.

Als syntaktische Strukturen der Sprache $L(r)$ erhalten wir so gewisse endliche angeordnete Bäume, deren Knoten mit Teilausdrücken des regulären Ausdrucks r markiert sind. Die Verkettung der Blättermarken eines Baumes aus $S(r)$ bildet ein Wort $w \in L(r)$.

Die Aufgabe der Syntaxanalyse besteht darin, zu vorgegebenem $w \in \Sigma^*$ diejenigen syntaktischen Strukturen aus $S(r)$ zu finden, deren Blättermarken dieses Wort w ergeben. Wie man das macht, behandeln wir in Kapitel 9.

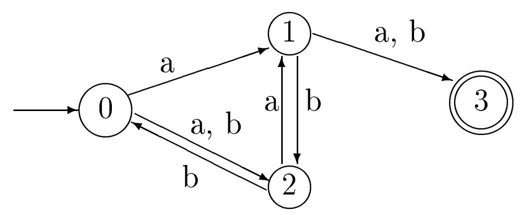
Aufgaben

Aufgabe 4.1 Man gebe einen endlichen, nichtdeterministischen Automaten $\mathcal{A}$ (mit ϵ-Übergängen) an, der die durch den Ausdruck $r = ((a+b)^* + ca)^*(ab + cba^*)^*$ definierte Sprache erkennt.

Aufgabe 4.2 Zu jedem deterministischen Automaten $\mathcal{A}$ (mit partieller Übergangsfunktion) gibt es einen vollständigen deterministischen Automaten $\mathcal{A}'$ (mit totaler Übergangsfunktion), der dieselbe Sprache erkennt und höchstens einen Zustand mehr hat.

Aufgabe 4.3

(i) Man gebe zum (partiellen) nicht-deterministischen Automaten $\mathcal{A}$ =



einen äquivalenten deterministischen Automaten $\mathcal{A}'$ an.

(ii) Konstruiere einen regulären Ausdruck für $L(\mathcal{A})$ nach der Methode der Zustands- bzw. Variablenelimination aus dem Satz von Kleene.

Aufgabe 4.4

(i) Zeige, daß L/B die größte Sprache $M \subseteq \Sigma^*$ ist, für die $M \cdot B \subseteq L$ gilt.

(ii) Nach Teil (i) ist stets $(L/B) \cdot B \subseteq L$. Gib ein Beispiel mit $(L/B) \cdot B \neq L$ an.

(iii) Zeige, daß $(L/L)^* = L/L$ für jedes $L \subseteq \Sigma^*$. Gib ein Beispiel $L, M \subseteq \Sigma^*$ mit $(L/M)^* \neq L/M$ an.

Aufgabe 4.5 (Substringsuche,Knuth-Morris-Pratt-Automat) (Vgl. Aho/Hopcroft/Ullman, S.330 ff)

Konstruiere zum Suchwort (oder Muster) $y := abbabbb =: b_1 \cdots b_7$ einen endlichen Automaten $\mathcal{A}_y$, mit dem man in einem Text $t \in \Sigma^*$ die Vorkommen von y suchen kann. Bei Eingabe von t erreiche $\mathcal{A}_y$ genau dann einen Endzustand, wenn ein Präfix der Form xy von t gelesen ist, d.h. $L(\mathcal{A}) = \Sigma^* y$. Das um weitere Übergänge zu ergänzende 'Skelett' von $\mathcal{A}_y$ ist

$$\longrightarrow 0 \xrightarrow{a} 1 \xrightarrow{b} 2 \xrightarrow{b} 3 \xrightarrow{a} 4 \xrightarrow{b} 5 \xrightarrow{b} 6 \xrightarrow{b} 7 \longrightarrow$$

a) Berechne zuerst die sogenannte *Fehlerfunktion* des Musters y,

$$f(j) := \begin{cases} \max\{\, s < j \mid b_1 \cdots b_s \equiv b_{j-s+1} \cdots b_j \,\}, & \text{falls es solche } s \text{ gibt,} \\ 0, & \text{sonst.} \end{cases}$$

b) Vervollständige das Skelett von $\mathcal{A}_y$ so, daß die Maschine wie folgt operiert:

Falls beim Lesen von $t = a_1 \cdots a_n$ nach dem Präfix $a_1 \cdots a_{k+j}$ der Automat $\mathcal{A}_y$ im Zustand j ist, sollte das Suffix $a_{k+1} \cdots a_{k+j}$ mit $b_1 \cdots b_j$ übereinstimmen, aber kein längeres Suffix von $a_1 \cdots a_{k+j}$ ein Präfix von y sein.

Falls $a_{k+j+1} \equiv b_{j+1}$ ist, geht $\mathcal{A}_y$ in den Zustand $j+1$. Im anderen Fall sagt die Fehlerfunktion, daß die letzten $f(j)$ Symbole von $a_1 \cdots a_{k+j}$ das nächstkürzere Suffix bilden, das mit einem Präfix des Musters y übereinstimmt, also mit $b_1 \cdots b_{f(j)}$. Falls $a_{k+j+1} \equiv b_{f(j)+1}$, so geht $\mathcal{A}_y$ in den Zustand $f(j)+1$ über, andernfalls testet man, ob $a_{k+j+1} \equiv b_{f(f(j))+1}$ ist, usw. Also ist

$$\delta(j,a) := \begin{cases} j+1, & \text{falls } a \equiv b_{j+1}, \\ 0, & \text{falls } j = 0 \text{ und } a \not\equiv b_1, \\ \delta(f(j),a) & \text{sonst.} \end{cases}$$

(Um die gefundenen Vorkommen von y in t anzugeben, muß man sich, wenn ein Endzustand erreicht ist, die Anzahl der gelesenen Symbole merken. Dazu bräuchte man zusätzlich einen Zähler und eine Liste der Länge $\leq |t|$.)

Aufgabe 4.6 (i) Definiere zum Text $t \in \Sigma^*$ die Äquivalenzrelation $\sim_t$ auf Σ^* durch

$$u \sim_t v \quad :\Longleftrightarrow \quad \forall w \in \Sigma^* \; (uw \in \mathit{Suffix}(t) \leftrightarrow vw \in \mathit{Suffix}(t)).$$

Gib einen Automaten $\mathcal{B}_t = (Q, \Sigma, \delta, q_0, F)$ zur Erkennung der Teilworte von t,

$$\mathit{Subw}(t) := \{\, v \in \Sigma^* \mid \exists u, w \in \Sigma^* \; t = uvw \,\},$$

an, dessen Zustandsmenge aus den Äquivalenzklassen $[u]_t := \{\, v \in \Sigma^* \mid u \sim_t v \,\}$ von $\sim_t$ besteht. Beweise $L(\mathcal{B}_t) = \mathit{Subw}(t)$.

(ii) Wie muß die Zustandsmenge F von $\mathcal{B}_t$ aus a) geändert werden, damit man einen (minimalen) Automaten zur Erkennung von $\mathit{Suffix}(t)$ erhält?

Aufgabe 4.7 Gib ein Verfahren zur Verringerung der Zustände von deterministischen endlichen Automaten an, das die Sprachäquivalenz von Zuständen 'von unten' approximiert, d.h. als wachsende Menge von Paaren äquivalenter Zustände berechnet.

Bemerkung: Es ist üblich, die Sprachäquivalenz $\sim_{\mathcal{A}} \subseteq Q \times Q$ 'von oben', d.h. als schrumpfende Menge von Paaren *nicht als nichtaquivalent erkannter* Zustände zu bestimmen. Die Nichtäquivalenz wird 'von unten', durch Relationen $I \subseteq \{ (p,q) \mid p \not\sim_{\mathcal{A}} q \}$, approximiert. I ergibt aber nur eine 'obere' Approximation $\sim_I := Q \times Q - I \supseteq \sim_{\mathcal{A}}$, die i.a. zu viele Zustände identifiziert, also einen wegen $L(A/\sim_I) \neq L(\mathcal{A})$ nutzlosen Äquivalenzklassenautomaten ergibt.

Im gewünschten Verfahren der Aufgabe berechnet man Approximationen $\sim \subseteq \sim_{\mathcal{A}}$, für die stets $L(\mathcal{A}/\sim) = L(\mathcal{A})$ ist. Daher kann man den minimalen Automaten durch eine Folge von äquivalenten Automaten mit schrumpfender Zustandszahl erreichen. Das kann auch dann praktisch durchführbar sein, wenn die übliche Berechnung an Zeit- oder Platzproblemen scheitert.

Aufgabe 4.8

(i) Zeige, daß die Sprache $L = \{a^{2n}b^n \mid n \in I\!N\}$ nicht regulär ist.

(ii) Zeige, daß zwar die Sprache

$$L_1 = \{a,b\}^* \cdot \{a^n b^n \mid n \in I\!N\} \cdot \{a,b\}^*,$$

nicht aber

$$L_2 = \{c,d\}^* \cdot \{a^n b^n \mid n \in I\!N\} \cdot \{c,d\}^*$$

regulär ist.

5 Endliche Automaten über Kleene'schen Algebren

Im Beweis des Satzes von Kleene haben wir gesehen, daß sich endliche Automaten als rechtslineare Gleichungssysteme auffassen lassen. Wir wollen nun die Automatentheorie ausführlicher als eine lineare Algebra über Kleene'schen Algebren darstellen.

5.1 Matrizen und rechtslineare Abbildungen

Definition 5.1 Sei $R = (R, +, 0, \cdot, 1)$ ein idempotenter Halbring. Eine Abbildung $f : R^n \to R^m$ heißt RECHTSLINEAR, falls für alle $x, y \in R^n$ und alle Elemente r aus R gilt:

$$f(x+y) = f(x) + f(y) \quad \text{und} \quad f(x \cdot r) = f(x) \cdot r.$$

Definition 5.2 Sei $R = (R, +, 0, \cdot, 1)$ ein idempotenter Halbring und $n, m \in I\!N$. Eine $m \times n$-MATRIX ÜBER R ist eine Abbildung $A : \{1, \ldots, m\} \times \{1, \ldots, n\} \to R$, die man gewöhnlich in der Form

$$A = \begin{pmatrix} a_{1,1} & \ldots & a_{1,n} \\ \vdots & & \vdots \\ a_{m,1} & \ldots & a_{m,n} \end{pmatrix}$$

darstellt, wobei $a_{i,j} = A(i,j)$ ist. Statt A schreibt man auch $(a_{i,j})$. Die Addition $A + B$ zweier $m \times n$-Matrizen A wird durch

$$\begin{pmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \cdots & a_{m,n} \end{pmatrix} + \begin{pmatrix} b_{1,1} & \cdots & b_{1,n} \\ \vdots & \ddots & \vdots \\ b_{m,1} & \cdots & b_{m,n} \end{pmatrix} := \begin{pmatrix} a_{1,1}+b_{1,1} & \cdots & a_{1,n}+b_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1}+b_{m,1} & \cdots & a_{m,n}+b_{m,n} \end{pmatrix}$$

erklärt, und die Multiplikation $A \cdot B$ einer $m \times n$-Matrix A mit einer $n \times r$-Matrix B durch

$$\begin{pmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \cdots & a_{m,n} \end{pmatrix} \cdot \begin{pmatrix} b_{1,1} & \cdots & b_{1,r} \\ \vdots & \ddots & \vdots \\ b_{n,1} & \cdots & b_{n,r} \end{pmatrix} := \begin{pmatrix} \Sigma_{k=1}^n a_{1,k}b_{k,1} & \cdots & \Sigma_{k=1}^n a_{1,k}b_{k,r} \\ \vdots & \ddots & \vdots \\ \Sigma_{k=1}^n a_{m,k}b_{k,1} & \cdots & \Sigma_{k=1}^n a_{m,k}b_{k,r} \end{pmatrix}.$$

Die Bezeichnungen 0_n und 1_n stehen für die $n \times n$-Matrizen

$$0_n := \begin{pmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{pmatrix} \qquad \text{und} \qquad 1_n := \begin{pmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{pmatrix}.$$

Die zu $A = (a_{i,j}) \in M_{m,n}(R)$ TRANSPONIERTE MATRIX ist

$$A^T := (a_{j,i}) = \begin{pmatrix} a_{1,1} & \ldots & a_{1,m} \\ \vdots & & \vdots \\ a_{n,1} & \ldots & a_{n,m} \end{pmatrix} \quad \in M_{n,m}(R).$$

Statt $M_{n,1}(R)$ schreiben wir $R^{(n)}$, für die aus Elementen $x^T = (x_1, \ldots, x_n) \in R^n$ gebildeten SPALTENVEKTOREN

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix},$$

und e_i für die Einheitsvektoren e_i mit $e_i^T = (0, \ldots, 1, \ldots, 0)$ mit 1 in der i-ten Komponente. Elemente $r \in R$ verstehen wir auch als 1×1-Matrizen. Damit sind insbesondere die Schreibweisen $A \cdot x$ und $x \cdot r$ erklärt.

Satz 5.3 Sei $(R, +, 0, \cdot, 1)$ ein idempotenter Halbring. Dann gelten:

(i) Eine Abbildung $f : R^n \to R^m$ ist genau dann rechtslinear, wenn es eine $m \times n$-Matrix A über R gibt, so daß $f(x) = A \cdot x$ für alle $x \in R^n$ ist.

(ii) Die rechtslinearen Abbildungen über R erfüllen mit den Operationen

$$(f + g)(x) := f(x) + g(x) \qquad \text{und} \qquad (f \cdot g)(x) := f(g(x))$$

die Axiome eines idempotenten Halbringes, wenn dabei die Dimensionen der Definitions- und Wertbereiche passen.

(iii) Die Menge $M_{n,n}(R)$ der $n \times n$-Matrizen über R bildet mit den oben erklärten Operationen einen idempotenten Halbring,

$$\mathcal{M}_{n,n}(R) = (M_{n,n}(R), +, 0_n, \cdot, 1_n).$$

Beweis: (i) Sei $f : R^n \to R^m$ rechtslinear. Für $1 \leq i \leq n$ sei $e_j \in R^n$ der j-te Spaltenvektor der $n \times n$-Matrix 1_n. Mit

$$f(e_j) = \begin{pmatrix} a_{1,j} \\ \vdots \\ a_{m,j} \end{pmatrix} \quad \text{setze} \quad A := \begin{pmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & & \vdots \\ a_{m,1} & \cdots & a_{m,n} \end{pmatrix}.$$

Dann ist für $x \in R^n$ mit $x_i \in R$ als i-ter Komponente wegen der Rechtslinearität von f

$$\begin{aligned} f(x) &= f(e_1 \cdot x_1 + \cdots + e_n \cdot x_n) \\ &= f(e_1) \cdot x_1 + \cdots + f(e_n) \cdot x_n \\ &= \begin{pmatrix} a_{1,1}x_1 \\ \vdots \\ a_{m,1}x_1 \end{pmatrix} + \cdots + \begin{pmatrix} a_{1,n}x_n \\ \vdots \\ a_{m,n}x_n \end{pmatrix} \\ &= A \cdot x. \end{aligned}$$

Ist umgekehrt A eine $m \times n$-Matrix über R, so sieht man aus den Definitionen, daß $A \cdot (x+y) = A \cdot x + A \cdot y$ und $A \cdot (x \cdot r) = (A \cdot x) \cdot r$ für alle $x, y \in R^n$ und $r \in R$. Also ist die durch $f(x) := A \cdot x$ definierte Abbildung $f : R^n \to R^m$ rechtslinear.

(ii) Das folgt daraus, daß die Operationen auf den rechtslinearen Funktionen punktweise auf die entsprechenden Operationen des Halbrings R zurückgeführt werden. Beispielsweise ist für $f : R^n \to R^m, g : R^m \to R^k, h : R^k \to R^l$ natürlich

$$((h \cdot g) \cdot f)(x) = h(g(f(x))) = (h \cdot (g \cdot f))(x)$$

und damit $(h \cdot g) \cdot f = h \cdot (g \cdot f)$.

(iii) Die Matrixoperationen $+$ und $\cdot$ sowie 1_n und 0_n sind die Übertragungen der Operationen aus (ii) und die Repräsentanten von $Id : R^n \to R^n$ und der alles auf $0 \in R^n$ werfenden Abbildung. Man prüfe nach, daß $A \cdot B$ die Abbildung $f \cdot g$ darstellt, wenn A und B die Abbildungen f und g darstellen; entsprechendes gilt für $A + B$ und $f + g$.

Für quadratische Matrizen einer festen Dimension n passen die Dimensionen der Definitions- und Wertebereiche stets überein, so daß die Behauptung aus (ii) folgt. □

Falls der zu Grunde liegende Ring $\mathcal{R}$ eine Kleene'sche Algebra ist, läßt sich auf dem Matrizenhalbring $\mathcal{M}_{n,n}(\mathcal{R})$ auch eine Iteration * definieren. Wenn M die rechtslineare Funktion $f : R^n \to R^n$ darstellt, stellt M^* die Iteration

$$f^* = \Sigma_{i=0}^{\infty} f^i$$

dar, analog zur Summenformel bei Relationen.

Lemma 5.4 Sei $\mathcal{K}$ eine Kleene'sche Algebra und $E \in \mathcal{M}_{n,n}(\mathcal{K})$. Es gibt eine eindeutig bestimmte Matrix $E^* \in \mathcal{M}_{n,n}(\mathcal{K})$ mit

$$EE^* + 1_n \leq E^* \quad \wedge \quad \forall X \in \mathcal{M}_{n,m}(K) \; (\, EX \leq X \to E^*X \leq X \,) \tag{21}$$
$$E^*E + 1_n \leq E^* \quad \wedge \quad \forall X \in \mathcal{M}_{m,n}(K) \; (\, XE \leq X \to XE^* \leq X \,). \tag{22}$$

Beweis: durch Induktion über n. Falls $n = 1$ ist, setzen wir $E^* = (e)^* := (e^*)$. Da $\mathcal{M}_{1,1}(\mathcal{K})$ isomorph zu $\mathcal{K}$ ist, wenn man (a) mit a identifiziert, folgt die Behauptung aus der Gültigkeit der Axiome *KA-1* und *KA-2* in $\mathcal{K}$.

Sei $n > 1$. Zerlege E in Untermatrizen A, B, C, D, mit quadratischen A und D, gemäß

$$E = \begin{pmatrix} A & B \\ C & D \end{pmatrix}.$$

Für ein ein $k < n$ ist $A \in \mathcal{M}_{k,k}(\mathcal{K})$ und $D \in \mathcal{M}_{n-k,n-k}(\mathcal{K})$, so daß nach Induktion schon Matrizen $A^* \in \mathcal{M}_{k,k}(\mathcal{K})$ und $D^* \in \mathcal{M}_{n-k,n-k}(\mathcal{K})$ mit den gewünschten Eigenschaften existieren. Ebenso gibt es schon $F^* \in \mathcal{M}_{k,k}(\mathcal{K})$ zu $F := A + BD^*C$. Definiere damit

$$E^* = \begin{pmatrix} A & B \\ C & D \end{pmatrix}^* = \begin{pmatrix} F^* & F^*BD^* \\ D^*CF^* & D^* + D^*CF^*BD^* \end{pmatrix}. \tag{23}$$

Zum Nachweis von $EE^* + 1_n \leq E^*$ genügt es, die entsprechenden Untermatrizen von $EE^* + 1_n$ und E^* zu vergleichen, d.h. zu zeigen:

$$\begin{aligned} AF^* + BD^*CF^* + 1_k &\leq F^* \\ AF^*BD^* + B(D^* + D^*CF^*BD^*) &\leq F^*BD^* \\ CF^* + DD^*CF^* &\leq D^*CF^* \\ CF^*BD^* + D(D^* + D^*CF^*BD^*) + 1_{n-k} &\leq D^*(CF^*BD^* + 1_{n-k}). \end{aligned}$$

Durch Ausklammern und mit $F = A + BD^*C$ erhalten wir die äquivalenten Ungleichungen

$$\begin{aligned} FF^* + 1_k &\leq F^* \\ (FF^* + 1_k)BD^* &\leq F^*BD^* \\ (DD^* + 1_{n-k})CF^* &\leq D^*CF^* \\ (DD^* + 1_{n-k})(CF^*BD^* + 1_{n-k}) &\leq D^*(CF^*BD^* + 1_{n-k}). \end{aligned}$$

Diese gelten wegen der Induktionsvoraussetzung über A^* und D^* und da nach 5.3 die Eigenschaften eines idempotenten Halbrings durch die Matrizen (mit passenden Dimensionen) erfüllt werden, also insbesondere die Matrixmultiplikation monoton ist.

Sei nun $X \in \mathcal{M}_{n,m}(\mathcal{K})$ mit $EX \leq X$. Zerlege X in Untermatrizen $x \in \mathcal{M}_{k,m}(\mathcal{K})$ und $y \in \mathcal{M}_{n-k,m}(\mathcal{K})$, so daß die Voraussetzung gleichwertig ist zu

$$Ax + By \leq x \quad \text{und} \quad Cx + Dy \leq y. \tag{24}$$

Um $E^*X \leq X$ zu zeigen, weisen wir die Ungleichungen

$$F^*x + F^*BD^*y \leq x \quad \text{und} \quad D^*CF^*x + (D^* + D^*CF^*BD^*)y \leq y \tag{25}$$

der entsprechenden Teilmatrizen nach. Aus (24) folgen

$$Ax \leq x, \; By \leq x, \; Cx \leq y \quad \text{und} \quad Dy \leq y,$$

und aus diesen nach den induktiv für A und D schon geltenden Behauptungen auch

$$A^*x \leq x \quad \text{und} \quad D^*y \leq y.$$

Mit der Monotonie der Matrixoperationen erhalten wir

$$Fx = Ax + (BD^*C)x \leq Ax + BD^*y \leq Ax + By \leq x,$$

woraus nach Induktion auch

$$F^*x \leq x$$

folgt. Damit haben wir $F^*x + F^*BD^*y \leq x + F^*x = x$, den ersten Teil von (25). Die für den zweiten Teil noch fehlenden Ungleichungen erhalten wir durch

$$D^*CF^*x \leq D^*Cx \leq D^*y \leq y \quad \text{und} \quad D^*CF^*BD^*y \leq D^*CF^*x \leq y.$$

Die entsprechenden Behauptungen über $E^*E + 1_n$ und XE zeigt man analog. □

Satz 5.5 (Kozen 1991) Ist $\mathcal{K}$ eine Kleene'sche Algebra, so ist auch $\mathcal{M}_{n,n}(\mathcal{K})$, mit der durch (23) definierten Iteration *, eine Kleene'sche Algebra.

Beweis: Von den Axiomen einer Kleene'schen Algebra fehlen uns nach Satz 5.3 (iii) und Lemma 5.4 nur noch die Implikationen aus den Axiomen *KA-1* und *KA-2*. Wähle nach der Voraussetzung in *KA-1* Matrizen $A, B, X \in \mathcal{M}_{n,n}(\mathcal{K})$ mit $AX + B \leq X$. Dann gelten $AX \leq X$ und $B \leq X$, so daß mit 5.4 und der Monotonie der Matrixmultiplikation auch

$$A^*B \leq A^*X \leq X$$

gilt, die Behauptung in *KA-1*. Entsprechend zeigt man *KA-2*. Also ist $\mathcal{M}_{n,n}(\mathcal{K})$ eine Kleene'sche Algebra. □

Da die Iteration eines Elements in einer Kleene'schen Algebra eindeutig bestimmt ist, hängt E^* auch nicht von der in der Konstruktion gewählten Zerlegung von E ab.

Endliche Automaten über einer Kleene'schen Algebra

Den Begriff eines endlichen Automaten $\mathcal{A} = (Q, I, \vdash_a, F)_{a\in\Sigma}$ über Σ verallgemeinern wir nun mit Hilfe der Matrizen. Ist $Q = \{q_1, \ldots, q_n\}$, so können wir die Übergangsrelationen $\vdash_a$ auf Q als 0-1-Matrizen A_a schreiben, wobei

$$A_a(i,j) = \begin{cases} 1, & \text{falls } q_i \vdash_a q_j \\ 0, & \text{sonst,} \end{cases}$$

und diese in einer einzigen Matrix $A = \sum_{a\in\Sigma} a \cdot A_a$ zusammenfassen. Die Anfangszustandsmenge I geben wir durch ihren 'charakteristischen Vektor'

$$u = \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix} \qquad \text{mit} \quad u_i = \begin{cases} 1, & \text{falls } q_i \in I \\ 0, & \text{sonst} \end{cases}$$

wieder, die Menge F der Endzustände durch einen weiteren charakteristischen Vektor, v. In der Iterierten A^* von A sollte dann im Feld (i,j) stehen, wie man durch mehrfache Verwendung der Übergangsmatrix A von Zustand q_i zu q_j kommt. Wir befreien uns von der speziellen Form der Matrix A.

Definition 5.6 Ein AUTOMAT $\mathcal{A} = \langle u, A, v\rangle$ ÜBER DER KLEENE'SCHEN ALGEBRA $\mathcal{K}$ besteht aus zwei Vektoren $u, v \in M_{n,1}(0,1)$ und einer Matrix $A \in \mathcal{M}_{n,n}(\mathcal{K})$. Das Element

$$L(\langle u, A, v\rangle) \;:=\; u^T A^* v \;\in \mathcal{K}$$

ist DIE VON $\mathcal{A}$ ERKANNTE SPRACHE, und $\{1, \dots, n\}$ die Zustandsmenge von $\mathcal{A}$.

Die von Σ FREI ERZEUGTE Kleene'sche Algebra $\mathcal{F}_\Sigma$ besteht aus der Menge der Äquivalenzklassen beweisbar gleicher regulärer Ausdrücke r über Σ, d.h. den

$$[r] := \{\, s \mid s \text{ ist regulärer Ausdrück über } \Sigma,\ KA \vdash r = s \,\},$$

zusammen mit den punktweise erklärten Operationen

$$[r] + [s] := [r + s], \quad [r] \cdot [s] := [r \cdot s], \quad [r]^* := [r^*].$$

Im folgenden werden wir meist nur endliche Automten über $\mathcal{F}_\Sigma$ betrachten.[5]

Beispiel 5.7 Die Übergangsrelationen $\vdash_a$ und $\vdash_b$ des Automaten $\mathcal{A}$ aus Beispiel 4.2 werden durch 3×3-Matrizen

$$M_a = a \cdot A_a = \begin{pmatrix} 0 & a & 0 \\ 0 & 0 & a \\ a & 0 & 0 \end{pmatrix} \quad \text{bzw.} \quad M_b = b \cdot A_b = \begin{pmatrix} 0 & 0 & b \\ 0 & b & 0 \\ 0 & b & 0 \end{pmatrix}$$

repräsentiert. Mit den Vektoren u, v für die Eingangs- und Endzustände erhalten wir

$$\langle u, A, v\rangle = \langle u, M_a + M_b, v\rangle = \langle \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 & a & b \\ 0 & b & a \\ a & b & 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \rangle$$

als Darstellung von $\mathcal{A}$. Die Iterierte A^* ergibt sich aus der Rekursionsformel (23) (wegen $0^* = 1$) wie folgt:

$$A^* = \left(\begin{array}{cc|c} 0 & a & b \\ 0 & b & a \\ \hline a & b & 0 \end{array}\right)^* = \left(\begin{array}{c|c} F^* & F^* B \\ \hline CF^* & 1 + CF^* B \end{array}\right)$$

mit

$$F = \begin{pmatrix} 0 & a \\ 0 & b \end{pmatrix} + \begin{pmatrix} b \\ a \end{pmatrix} \cdot (\, a \quad b\,) = \begin{pmatrix} 0 & a \\ 0 & b \end{pmatrix} + \begin{pmatrix} ba & bb \\ aa & ab \end{pmatrix} = \begin{pmatrix} ba & a + bb \\ aa & b + ab \end{pmatrix}.$$

[5] Nach dem Vollständigkeitssatz der Prädikatenlogik ist $r = s$ in KA genau dann beweisbar, wenn für jede Kleene'sche Algebra $\mathcal{K}$ und jede Wahl der $a^{\mathcal{K}} \in K$, $a \in \Sigma$, die Elemente $r^{\mathcal{K}}$ und $s^{\mathcal{K}}$ dieselben sind.

Für F^* ergibt die Formel

$$F^* = \begin{pmatrix} ba & a+bb \\ aa & b+ab \end{pmatrix}^* = \begin{pmatrix} f^* & f^*(a+bb)(b+ab)^* \\ (b+ab)^*aaf^* & (b+ab)^*(1+aaf^*(a+bb)(b+ab)^*) \end{pmatrix},$$

mit $f = ba + (a+bb)(b+ab)^*aa$. Wir erhalten für die erkannte Sprache

$$L(\langle u, A, v\rangle) \;=\; u^T A^* v \;=\; (1 \quad 0 \quad 0) \cdot A^* \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix},$$

also den Eintrag von A^* im Feld $(1,2)$, was hier das Feld $(1,2)$ von F^* ist, d.h.

$$f^*(a+bb)(b+ab)^* = (ba + (a+bb)(b+ab)^*aa)^*(a+bb)(b+ab)^* \;\; =: r \in K.$$

Ist $\mathcal{K} = \mathcal{F}_{\{a,b\}}$, so ist r gerade der erwartete reguläre Ausdruck mit $L = L(r) = L(\mathcal{A})$.

5.2 Die Vollständigkeit der Kleene'schen Algebra

Wir beweisen nun, der Darstellung in Kozen[12] folgend, den Vollständigkeitssatz von D. Kozen über die Axiomatisierung der Iteration: jede Gleichung $r = s$ zwischen regulären Ausdrücken, die bei der Sprachinterpretation wahr ist, kann in der Theorie der Kleene'schen Algebra bewiesen werden.

Dazu werden die Übersetzung regulärer Ausdrücke in endliche Automaten, die Umwandlung nicht-deterministischer in deterministische endliche Automaten und die Minimierung der Zustandsanzahl deterministischer endlicher Automaten aus Abschnitt 4.3 in der Theorie KA durch algebraische Umwandlungen mit Matrizen über $\mathcal{F}_\Sigma$ nachvollzogen. Reguläre Ausdrücke r über Σ identifizieren wir im Folgenden mit ihrer Bedeutung $[r] := \{\, s \mid KA \vdash r = s \,\}$ in $\mathcal{F}_\Sigma$.

Einfache ε-freie Automaten über $\mathcal{F}_\Sigma$

Um die Argumente aus Abschnitt 4.3 in $\mathcal{F}_\Sigma$ nachvollziehen zu können, brauchen wir Automaten $\langle u, A, v\rangle$ über $\mathcal{F}_\Sigma$, deren Übergangsmatrix A nicht beliebige reguläre Ausdrücke enthält, sondern nur Summen von Buchstaben $a \in \Sigma$.

Definition 5.8 Ein Automat $\langle u, A, v\rangle$ über $\mathcal{F}_\Sigma$ heißt EINFACH, wenn die Matrix A als Summe

$$A = J + \sum_{a \in \Sigma} a \cdot A_a \tag{26}$$

mit $0,1$-Matrizen J und A_a dargestellt werden kann. $\mathcal{A}$ heißt ε-FREI, wenn J die Nullmatrix ist.

Man überlege sich, daß einfache, ε-freie Automaten den früher verwendeten Automaten entsprechen. Die früher nicht verwendeten SPONTANEN oder ε-ÜBERGÄNGE kann man wie folgt beseitigen:

Lemma 5.9 Zu jedem einfachen Automaten $\mathcal{A} = \langle u, A, v\rangle$ gibt es einen einfachen, ε-freien Automaten $\mathcal{B} = \langle s, B, t\rangle$ mit $u^T A^* v = s^T B^* t$.

Beweis: Nach Voraussetzung kann A mit $0, 1$-Matrizen J und A_a in der Form

$$A = J + \sum_{a\in\Sigma} a \cdot A_a = J + A'$$

dargestellt werden. Nach Proposition 3.10 ist

$$u^T A^* v = u^T (A' + J)^* v = u^T J^* (A' J^*)^* v.$$

Da A' schon einfach und ε-frei und J^* eine $0, 1$-Matrix ist, ist $A'J^*$ einfach und ε-frei, weshalb

$$\langle s, B, t\rangle := \langle (u^T J^*)^T, A' J^*, v\rangle$$

das Gewünschte leistet. □

Übersetzung regulärer Ausdrücke in endliche Automaten

Als Nächstes zeigen wir, daß der Darstellungssatz 4.5 von Kleene in KA beweisbar ist. Es genügt, zu jedem regulären Ausdruck r einen einfachen Automaten über $\mathcal{F}_\Sigma$ zu konstruieren, da ε-Übergänge mit 5.9 anschließend beseitigt werden können.

Satz 5.10 Zu jedem regulären Ausdruck r über Σ gibt es einen einfachen Automaten $\mathcal{A}_r = \langle u, A, v\rangle$ über $\mathcal{F}_\Sigma$ mit $r = u^T A^* v$.

Beweis: durch Induktion über den Aufbau von r.

$r \equiv 0$: Mit $\mathcal{A}_0 = \langle (\,0\,), (\,0\,), (\,0\,)\rangle$ ist $(0)(0)^*(0) = 00^*0 = 0$.

$r \equiv 1$: Mit $\mathcal{A}_1 = \langle (\,1\,), (\,0\,), (\,1\,)\rangle$ ist $(1)(0)^*(1) = 0^* = 1$.

$r \equiv a$ für ein $a \in \Sigma$: Sei

$$\mathcal{A}_a = \langle \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0 & a\\0 & 0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}\rangle.$$

Dann ist

$$L(\mathcal{A}) = (1,0) \cdot \begin{pmatrix}0 & a\\0 & 0\end{pmatrix}^* \cdot \begin{pmatrix}0\\1\end{pmatrix} = (1,0) \cdot \begin{pmatrix}1 & a\\0 & 1\end{pmatrix} \cdot \begin{pmatrix}0\\1\end{pmatrix} = a.$$

$r \equiv r_1 + r_2$: Seien $\mathcal{A}_{r_1} = \langle u, A, v\rangle$ und $\mathcal{A}_{r_2} = \langle s, B, t\rangle$, so daß induktiv schon

$$r_1 = u^T A^* v \quad \text{und} \quad r_2 = s^T B^* t$$

gelten. Um die Automaten parallelzuschalten, machen wir zunächst die Zustandsmengen, d.h. die Zeilen- und Spaltenindizes, disjunkt, indem wir A und B mit Nullmatrizen passender Dimensionen erweitern und zur Übergangsmatrix

$$\begin{pmatrix} A & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & B \end{pmatrix} = \begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix}$$

zusammenfassen. Dann vereinigen wir die Mengen der Anfangs- und Endzustände, was zu

$$\mathcal{A}_{r_1+r_2} := \langle \begin{pmatrix} u \\ s \end{pmatrix}, \begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix}, \begin{pmatrix} r \\ t \end{pmatrix} \rangle$$

führt. Mit der der Rekursionsformel für die Iterierte einer Matrix erhalten wir

$$\begin{aligned} L(\mathcal{A}_{r_1+r_2}) &= \begin{pmatrix} u \\ s \end{pmatrix}^T \cdot \begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix}^* \cdot \begin{pmatrix} r \\ t \end{pmatrix} \\ &= \begin{pmatrix} u & s \end{pmatrix} \cdot \begin{pmatrix} A^* & 0 \\ 0 & B^* \end{pmatrix} \cdot \begin{pmatrix} r \\ t \end{pmatrix} \\ &= u^T A^* v + s^T B^* t \\ &= r_1 + r_2. \end{aligned}$$

$r \equiv r_1 \cdot r_2$: Seien wieder $\mathcal{A}_{r_1} = \langle u, A, v\rangle$ und $\mathcal{A}_{r_2} = \langle s, B, t\rangle$, so daß induktiv schon

$$r_1 = u^T A^* v \quad \text{und} \quad r_2 = s^T B^* t$$

gelten. Um die Automaten hintereinanderzuschalten, werden die Endzustände des ersten durch ε-Übergängen mit den Anfangszuständen des zweiten verbunden; als Anfanszustände sind die des ersten und als Endzustände die des zweiten zu nehmen. Das ergibt

$$\mathcal{A}_{r_1 \cdot r_2} := \langle \begin{pmatrix} u \\ 0 \end{pmatrix}, \begin{pmatrix} A & vs^T \\ 0 & B \end{pmatrix}, \begin{pmatrix} 0 \\ t \end{pmatrix} \rangle.$$

Die davon erkannte Sprache ist

$$\begin{aligned} L(\mathcal{A}_{r_1 \cdot r_2}) &= \begin{pmatrix} u \\ 0 \end{pmatrix}^T \cdot \begin{pmatrix} A & vs^T \\ 0 & B \end{pmatrix}^* \cdot \begin{pmatrix} 0 \\ t \end{pmatrix} \\ &= \begin{pmatrix} u & 0 \end{pmatrix} \cdot \begin{pmatrix} A^* & A^* vs^T B^* \\ 0 & B^* \end{pmatrix} \cdot \begin{pmatrix} 0 \\ t \end{pmatrix} \\ &= u^T A^* v s^T B^* t \\ &= r_1 \cdot r_2. \end{aligned}$$

$r \equiv s^*$: Sei $\mathcal{A}_s = \langle u, A, v\rangle$ mit $s = u^T A^* v$. Mehrfache Rückkopplung von $\mathcal{A}$ wird dadurch erreicht, daß wir die Endzustände durch ε-Übergänge mit den Anfangszuständen verbinden. Das ergibt den Automaten

$$\mathcal{A}_{ss^*} := \langle u, A + vu^T, v\rangle,$$

der –mit Proposition 3.10 und Aufgabe 3.2 b) (iv)– die Sprache

$$\begin{aligned} L(\mathcal{A}_{ss^*}) &= u^T \cdot (A + vu^T)^* \cdot v \\ &= u^T A^* (vu^T A^*)^* v \\ &= u^T A^* v (u^T A^* v)^* \\ &= ss^* \end{aligned}$$

erkennt. Als Automaten $\mathcal{A}_{s^*}$ nehmen wir dann $\mathcal{A}_{ss^*+1}$. □

Umwandlung in deterministische endliche Automaten

Ein Automat $\mathcal{A} = \langle u, A, v\rangle$ heißt DETERMINISTISCH, wenn er einfach und ε-frei ist, also $A = \sum_{a\in\Sigma} aA_a$ für 0-1-Matrizen A_a, und sowohl u wie alle Zeilen der A_a genau eine 1 enthalten.

Lemma 5.11 Für jeden einfachen, ε-freien Automaten $\mathcal{A} = \langle u, A, v\rangle$ über $\mathcal{F}_\Sigma$ gibt es einen deterministischen Automaten $\widehat{\mathcal{A}} = \langle \widehat{u}, \widehat{A}, \widehat{u}\rangle$ über $\mathcal{F}_\Sigma$ mit $u^T A^* v = \widehat{u}^T \widehat{A}^* \widehat{v}$.

Beweis: Sei $A \in M_{n,n}(\mathcal{F}_\Sigma)$. Wir konstruieren eine lineare Projektion $X^T : {\mathcal{F}_\Sigma}^{(2^n)} \to {\mathcal{F}_\Sigma}^{(n)}$, so daß folgendes Diagramm kommutiert:

$$\begin{tikzcd} {\mathcal{F}_\Sigma}^{(n)} \arrow[r, "A"] \arrow[d, dotted, "X"'] & {\mathcal{F}_\Sigma}^{(n)} \arrow[d, dotted, "X"] \\ {\mathcal{F}_\Sigma}^{(2^n)} \arrow[r, dotted, "\widehat{A}"] & {\mathcal{F}_\Sigma}^{(2^n)} \end{tikzcd}$$

Jeder Teilmenge S der 'Zustandsmenge' $\{1, 2, \ldots, n\}$ von $\mathcal{A}$ entspricht der $0,1$-Vektor $s \in M_{n,1}(\mathcal{F}_\Sigma)$, dessen i-te Komponente genau dann 1 ist, wenn i in der Teilmenge S liegt. Andererseits können wir diesen Vektor als die Binärdarstellung einer Zahl $s \in \{1, \ldots, 2^n\}$ lesen und ihm den 0,1-Vektor $e_s \in M_{2^n,1}(\mathcal{F}_\Sigma)$ zuordnen, der nur an der s-ten Komponente eine 1 hat.

Sei X^T die $n \times 2^n$-Matrix mit $X^T e_s = \sum_{i \in s} e_i = s$, also X die $2^n \times n$-Matrix, deren s-te Zeile gerade s^T ist, d.h.

$$e_s^T X = s^T.$$

für alle $s \in M_{n,1}$. Nach der Voraussetzung kann A mit 0, 1-Matrizen A_a in der Form

$$A = \sum_{a \in \Sigma} a \cdot A_a$$

dargestellt werden. Für jedes $a \in \Sigma$ sei $\widehat{A}_a$ die $2^n \times 2^n$-Matrix mit $e^T_{(s^T A_a)^T}$ als s-ter Zeile, also

$$e_s^T \widehat{A}_a = e^T_{(s^T A_a)^T}.$$

Dann ist für alle n-zeiligen 0-1-Vektoren s

$$e_s^T X A_a \;=\; s^T A_a \;=\; (s^T A_a)^{TT} \;=\; e^T_{(s^T A_a)^T} X \;=\; e_s^T \widehat{A}_a X,$$

also $XA_a = \widehat{A}_a X$. Der Automat

$$\langle \hat{u}, \widehat{A}, \hat{v} \rangle := \langle e_u, \textstyle\sum_{a \in \Sigma} a \cdot \widehat{A}_a, Xv \rangle.$$

ist nach Konstruktion einfach und ε-frei, und deterministisch, da e_u und alle Zeilen der $\widehat{A}_a$ jeweils nur eine1 enthalten. Mit $Xa = aX$ folgt $XA = \widehat{A}X$, und daraus nach Aufgabe 3.2 b) auch $XA^* = \widehat{A}^* X$. Also gilt

$$\hat{u}^T \widehat{A}^* \hat{v} \;=\; e_u^T \widehat{A}^* X v \;=\; e_u^T X A^* v \;=\; u^T A^* v,$$

und die beiden Automaten erkennen dieselbe Sprache. □

Minimierung der Zustandsanzahl

Ein Element $a \in K$ heißt INVERTIERBAR, wenn es sowohl links- wie rechtsinvertierbar ist, d.h. Elemente $b, c \in K$ mit $ba = 1 = ac$ gibt. Diese sind wegen $b = b(ac) = (ba)c = c$ gleich, folglich eindeutig bestimmt, und werden mit a^{-1} bezeichnet. (Im allgemeinen gibt es mehrere Links- und mehrere Rechtsinverse. Ist aber b ein linksinvertierbares Linksinverses von a, so ist es auch ein Rechtsinverses von a: aus $cb = 1 = ba$ folgt $ab = 1$. Denn $ab = 1ab = (cb)(ab) = c(ba)b = c1b = cb = 1$.)

Proposition 5.12 Sei $\mathcal{K}$ eine Kleene'sche Algebra und $p \in K$ ein invertierbares Element. Dann ist für jedes $a \in K$

$$(pap^{-1})^* = pa^*p^{-1}.$$

Beweis: Nach Aufgabe 3.2 b) ist $(pq)^*p = p(qp)^*$ für alle $q \in K$, und daher $(pq)^* = (pq)^*pp^{-1} = p(qp)^*p^{-1}$. Mit $q = ap^{-1}$ folgt die Behauptung. □

Die Vertauschung von Zuständen eines Automaten ändert also die erkannte Sprache nicht:

Folgerung 5.13 Sei $\mathcal{A} = \langle u, A, v\rangle$ mit $A \in M_{n,n}(\mathcal{F}_\Sigma)$ ein Automat, P die repräsentierende Matrix einer Permutation π, d.h. $Pe_i = e_{\pi(i)}$ für alle $i \in \{1, \ldots, n\}$, und $\mathcal{B} = \langle Pu, PAP^T, Pv\rangle$. Da P invertierbar ist, mit Inverse P^T, ist

$$L(\mathcal{B}) = (Pu)^T(PAP^T)^*(Pv) = (u^TP^T)(PA^*P^T)(Pv) = u^TA^*v.$$

Ein Zustand q von $\langle u, A, v\rangle$ ist ERREICHBAR, wenn $u^TA^*e_q \neq 0$ ist.

Lemma 5.14 Zu jedem deterministischen Automaten $\mathcal{A} = \langle u, A, v\rangle$ gibt es einen äquivalenten deterministischen Automaten $\langle u_R, A_R, v_R\rangle$, der nur erreichbare Zustände hat.

Beweis: Sei $Q = \{1, \ldots, n\}$ die Menge der Zustände von A, R die Menge der erreichbaren, $U = Q - R$ die der unerreichbaren. Wir können annehmen, daß $R = \{1, \ldots, r\}$ und $U = \{r + 1, \ldots, n\}$, da wir sonst mit 5.13 eine entsprechende Vertauschung vornehmen. Die Matrix A und die Vektoren U, v haben dann die Form

$$A = \begin{pmatrix} A_{RR} & A_{RU} \\ A_{UR} & A_{UU} \end{pmatrix}, \quad u = \begin{pmatrix} u_R \\ u_U \end{pmatrix} \quad \text{und} \quad v = \begin{pmatrix} v_R \\ v_U \end{pmatrix}.$$

Hierin müssen $A_{RU} = 0$ und $u_U = 0$ sein, da sonst unerreichbare Zustände erreichbar wären. Nach

$$\begin{aligned} u^TA^*v &= (\, u_R^T \quad 0\,) \cdot \begin{pmatrix} A_{RR} & 0 \\ A_{UR} & A_{UU} \end{pmatrix}^* \cdot \begin{pmatrix} v_R \\ v_U \end{pmatrix} \\ &= (\, u_R^T \quad 0\,) \cdot \begin{pmatrix} A_{RR}^* & 0 \\ A_{UU}^*A_{UR}A_{RR}^* & A_{UU}^* \end{pmatrix} \cdot \begin{pmatrix} v_R \\ v_U \end{pmatrix} \\ &= u_R^TA_{RR}^*v_R \end{aligned}$$

erkennen die Automaten dieselbe Sprache. □

In einem deterministischen Automaten $\langle u, A, v\rangle$ gibt es zu jedem Zustand p und jedem $a \in \Sigma$ genau ein q mit $A_a(p, q) = 1$; dieses q werde mit $\delta(p, a)$ bezeichnet. Eine Äquivalenzrelation $\equiv$ auf der Zustandsmenge heißt MYHILL-NERODE'sch, wenn

$$\delta(p, a) \equiv \delta(q, a) \quad \text{and} \quad e_p^Tv = e_q^Tv$$

für alle Zustände $p \equiv q$ und alle $a \in \Sigma$.

Lemma 5.15 Sei $\mathcal{A} = \langle u, A, v\rangle$ ein deterministischer Automat, dessen n Zustände alle erreichbar sind. Sei $\equiv$ eine Myhill-Nerode Äquivalenzrelation auf $\{1, \ldots, n\}$ mit m Äquivalenzklassen. Es gibt einen deterministischen Automaten $\overline{\mathcal{A}} := \langle \overline{u}, \overline{A}, \overline{v}\rangle$ mit m Zuständen und

$$u^TA^*v = \overline{u}^T\overline{A}^*\overline{v}.$$

Beweis: Sei $A = \sum_{a\in\Sigma} a \cdot A_a$ mit 0-1-Matrizen A_a. Wir konstruieren eine lineare Projektion $Y : \mathcal{F}_\Sigma{}^{(m)} \to \mathcal{F}_\Sigma{}^{(n)}$, so daß folgendes Diagramm kommutiert:

$$\begin{tikzcd}
\mathcal{F}_\Sigma{}^{(n)} \arrow[r, "A"] & \mathcal{F}_\Sigma{}^{(n)} \\
\mathcal{F}_\Sigma{}^{(m)} \arrow[u, dotted, "Y"] \arrow[r, dotted, "\overline{A}"] & \mathcal{F}_\Sigma{}^{(m)} \arrow[u, dotted, "Y"']
\end{tikzcd}$$

Zu $p \in Q := \{1, \ldots, n\}$ sei $[p] = \{\, q \in Q \mid p \equiv q \,\}$. Sei Y die durch $Ye_{[p]} = \sum_{q\in[p]} e_q$ definierte 0-1-Matrix der Dimension $n \times m$, so daß also

$$e_p^T Y = e_{[p]}^T.$$

Für jedes $a \in \Sigma$ sei $\overline{A}_a$ die $m \times m$-Matrix mit

$$e_{[p]}^T \overline{A}_a = e_{[\delta(p,a)]}^T.$$

Dann ist

$$e_p^T A_a Y = e_{\delta(p,a)}^T Y = e_{[\delta(p,a)]}^T = e_{[p]}^T \overline{A}_a = e_p^T Y \overline{A}_a.$$

Also ist $A_a Y = Y\overline{A}_a$, und mit $aY = Ya$ auch $AY = Y\overline{A}$ für

$$\overline{A} = \sum_{a\in\Sigma} a \cdot \overline{A}_a.$$

Seien $\overline{u} = Y^T u$ und $\overline{v}$ der k-zeilige 0-1-Vektor mit $e_{[p]}^T \overline{v} = e_p^T v$. Da wegen $AY = Y\overline{A}$ auch $A^*Y = Y\overline{A}^*$ ist, erhalten wir

$$\overline{u}^T \overline{A}^* \overline{v} = u^T Y \overline{A}^* \overline{v} = u^T A^* Y \overline{v} = u^T A^* v,$$

die Äquivalenz von $\overline{\mathcal{A}}$ und $\mathcal{A}$. □

Damit können wir zeigen, daß die Theorie der Kleene'schen Algebra alle Gleichungen, die bei der Sprachinterpretation gelten, umfaßt:

Satz 5.16 (Vollständigkeitssatz für *KA*, Kozen 1991) Seien r_1 und r_2 reguläre Ausdrücke über Σ mit $L(r_1) = L(r_2)$. Dann ist $r_1 = r_2$ in *KA* beweisbar.

Beweis: Seien $\mathcal{A} = \langle u, A, v\rangle$ und $\mathcal{B} = \langle s, B, t\rangle$ minimale, deterministische endliche Automaten über $\mathcal{F}_\Sigma$ mit

$$L(r_1) = L(u^T A^* v) \quad \text{und} \quad L(r_2) = L(s^T B^* t).$$

Nach den vorangegangenen Lemmata sind dann

$$r_1 = u^T A^* v \quad \text{und} \quad r_2 = s^T B^* t$$

in der Theorie KA beweisbar. Da $L(r_1) = L(r_2)$ ist, sind diese Automaten isomorph, also bis auf eine Bijektion der Zustandsmengen gleich. Sei $P \in M_{n,n}(\mathcal{F}_\Sigma)$ eine 0-1-Matrix, die diesen Isomorphismus repräsentiert, also

$$\langle s, B, t \rangle = \langle Pu, PAP^T, Pv \rangle.$$

Dann ist mit Proposition 5.12

$$r_2 \;=\; s^T B^* t \;=\; (Pu)^T (PAP^T)^* (Pv) \;=\; (u^T P^T)(PA^*P^T)(Pv) \;=\; u^T A^* v \;=\; r_1.$$

Damit ist $r_1 = r_2$ in KA beweisbar. □

5.3 Kleene'sche Halbmoduln

Die Methoden der linearen Algebra, die im vorigen Abschnitt zur Darstellung der regulären Sprachen benutzt wurden, lassen sich noch etwas allgemeiner verwenden und auf Sprachklassen anwenden, die nicht unter Verkettung und Iteration abgeschlossen sind.

Dazu betrachten wir Bereiche mit einer Addition, deren Elemente durch eine geeignete "äußere" Multiplikation : mit Elementen einer Kleene'schen Algebra multipliziert werden können.

Definition 5.17 Ein (linker) HALBMODUL $\mathcal{M} = (\mathcal{R}, \mathcal{A}, :)$ besteht aus einer Kleene'schen Algebra $\mathcal{R} = (R, +, \cdot, ^*, 0, 1)$, einer idempotenten kommutativen Halbgruppe $\mathcal{A} = (A, +, 0)$ und einer SKALARMULTIPLIKATION : von $R \times A$ nach A, so daß für alle $r, s \in R$ und $a, b \in A$ gilt:

$$\begin{array}{rcl@{\qquad}rcl} r:0 &=& 0, & r:(a+b) &=& r:a + r:b, \\ 0:a &=& 0, & (r+s):a &=& r:a + s:a, \\ 1:a &=& a, & (r\cdot s):a &=& r:(s:a). \end{array}$$

Abkürzend nennen wir auch einfach $(\mathcal{A}, :) = (A, +, 0, :)$ einen HALBMODUL ÜBER $\mathcal{R}$. Falls zusätzlich für alle $r \in R$ und $a, b \in A$ auch

$$r : b + a \leq b \;\rightarrow\; r^* : a \leq b \tag{27}$$

gilt, nennen wir $\mathcal{M}$ einen (linken) KLEENE'SCHEN HALBMODUL.

Aus den Distributiveigenschaften folgt, daß die Skalarmultiplikation in beiden Argumenten monoton ist. Insbesondere ist $r^n : a \leq r^* : a$.

Beispiel 5.18 Ist $\mathcal{R}$ eine Kleene'sche Algebra, so ist $\mathcal{M} = (\mathcal{R}, \mathcal{A}, \cdot)$ mit $\mathcal{A} = (R, +, 0)$ ein Kleene'scher Halbmodul über $\mathcal{R}$, wobei $+$ und $\cdot$ die jeweilige Operation von $\mathcal{R}$ ist.

Proposition 5.19 In einem Kleene'schen Halbmodul $(\mathcal{R}, \mathcal{A}, :)$ hat die rechtslineare Gleichung $x = r : x + a$, mit $r \in R, a \in A$, eine kleinste Lösung, nämlich $x = r^* : a$.

Beweis: Mit den Eigenschaften einer Kleene'schen Algebra gilt

$$r^* : a \;=\; (r \cdot r^* + 1) : a \;=\; (r \cdot r^*) : a + 1 : a \;=\; r : (r^* : a) + a,$$

so daß $r^* : a$ eine Lösung von $x = r : x + a$ ist. Jede weitere Lösung b ist nach (27) mindestens so groß wie $r^* : a$. □

Hauptsächlich haben wir folgende Moduln im Sinn:

Beispiel 5.20 Wir geben nur $\cdot$ und $:$ an, da * damit durch die Summenformel $R^* = \bigcup\{ R^n \mid n \in I\!N \}$ festgelegt ist.

(i) (C. S. Peirce, 1875) Der Halbmodul $\mathcal{M}$ bestehe aus

$$\mathcal{R} = (2^{M \times M}, \cup, \circ, ^*, \emptyset, 1_M) \quad \text{und} \quad \mathcal{A} = (2^M, \cup, \emptyset),$$

also der Algebra der zweistelligen Relationen auf einer Menge M mit der Algebra der Teilmengen von M als Modul, wobei $\cdot$ und $:$ durch

$$R \circ S \;=\; \{ (m,n) \mid \exists k \in M \; (R(m,k) \wedge S(k,n)) \}. \tag{28}$$

$$R : A \;=\; \{ m \mid \exists a \in A \; R(m,a) \} \tag{29}$$

gegeben sind, also die Relationenkomposition und das UrBild von Mengen unter Relationen. Die Summenformel für R^* ergibt die reflexive transitive Hülle von R.

(ii) (J. Gruska, 1971) Der Halbmodul $\mathcal{M}$ bestehe aus

$$\mathcal{R} = (2^{\Sigma^* \times \Sigma^*}, \cup, \cdot, ^*, \emptyset, \{(\varepsilon, \varepsilon)\}) \quad \text{und} \quad \mathcal{A} = (2^{\Sigma^*}, \cup, \emptyset),$$

also den binären Wortrelationen über einem Alphabet Σ und den Wortmengen über Σ als Modul, wobei $\cdot$ und $:$ durch

$$R \cdot S \;=\; \{ (v_1 w_1, w_2 v_2) \mid (v_1, v_2) \in R, (w_1, w_2) \in S \}$$

$$R : A \;=\; \{ v_1 w v_2 \mid (v_1, v_2) \in R, w \in A \},$$

die (elementweise) Einfügung einer Wortrelation bzw. -menge in eine Wortrelation interpretiert werden.

(iii) (C. Pollard[20], 1984) Zur Beschreibung natürlicher Sprachen wurden neben $\cdot$ aus (ii) die folgenden 'wrapping'-Operationen $\circ_i : \Sigma^{*2} \times \Sigma^{*2} \to \Sigma^{*2}$ (auch c-ins, cat_1, cat_2, l-ins, r-ins genannt) untersucht:

$$\begin{array}{lcllcl}
(v_1,v_2) \circ_1 (w_1,w_2) & := & (v_1w_1, w_2v_2) & & & \\
(v_1,v_2) \circ_2 (w_1,w_2) & := & (v_1, v_2w_1w_2) & (v_1,v_2) \circ_3 (w_1,w_2) & := & (v_1v_2w_1, w_2) \\
(v_1,v_2) \circ_4 (w_1,w_2) & := & (v_1, w_1w_2v_2) & (v_1,v_2) \circ_5 (w_1,w_2) & := & (v_1w_1w_2, v_2).
\end{array}$$

Wie in (ii) ergibt jede der drei assoziativen Operationen $\circ_1, \circ_2, \circ_3$, auf Mengen fortgesetzt, eine Kleene'schen Algebra $\mathcal{R}_i = (2^{\Sigma^* \times \Sigma^*}, \cup, \circ_i, ^*, \emptyset, \{(\varepsilon,\varepsilon)\})$ (vgl. 3.11).

Seien $\circ_i'$ die entsprechenden inhomogene Versionen dieser Operationen, bei denen ein Argument ein Wort und das andere ein Wortpaar ist:

$$\begin{array}{lcllcl}
(v_1,v_2) \circ_1' w & := & v_1wv_2 & & & \\
(v_1,v_2) \circ_2' w & := & (v_1, v_2w) & v \circ_3' (w_1,w_2) & := & (vw_1, w_2) \\
(v_1,v_2) \circ_4' w & := & (v_1, wv_2) & (v_1,v_2) \circ_5' w & := & (v_1w, v_2)
\end{array}$$

Wegen $A \circ' (A \circ' X) = (A \circ A) \circ' X$ ist $\mathcal{A} = (2^{\Sigma^*}, \cup, \emptyset)$ für die assoziativen Operationen $\circ \in \{\circ_1, \circ_2, \circ_3\}$ ebenfalls ein (linker bzw. rechter) Modul über $\mathcal{R}_i$.

(iv) Varianten $\mathcal{L}_\Sigma^n$ der Sprachinterpretation $\mathcal{L}_\Sigma$ erhält man durch Anheben von

$$(a_1, \ldots, a_n) \cdot (b_1, \ldots, b_n) := (a_1b_1, \ldots, a_nb_n)$$

auf n-stellige Relationen zwischen Wörtern. Dies sind Kleene'sche Algebren, und mit $\cdot$ als Skalarprodukt auch Kleene'sche Halbmoduln über sich. Damit läßt sich z.B. $\{\, a^nb^nc^n \mid n \in I\!N \,\}$ als Kollaps der regulären Wortrelation

$$R^* = \mu X.(\{(a,b,c)\} \cdot X + 1) = \{\, (a^n, b^n, c^n) \mid n \in I\!N \,\}$$

darstellen, wobei $\mu X.(A \cdot X + 1)$ die kleinste Lösung von $X = A \cdot X + 1$ in $\mathcal{L}_\Sigma^3$ meint.

Lemma 5.21 Die Halbmoduln aus 5.20 sind Kleene'sche Halbmoduln, so daß jeweils $R^* : A$ die kleinste Lösung von $x = R : x + A$ ist.

Beweis: Es ist nicht schwer, die Eigenschaften eines Halbmoduls zu überprüfen. Zum Nachweis von (27) bei (i) sei $R : B \cup A \subseteq B$, nach den Halbmoduleigenschaften also $R^n : A \subseteq B$ für jedes n. Für $R^* : A \subseteq B$ genügt daher zu zeigen, daß

$$R^* : A = \bigcup \{\, R^k : A \mid k \in I\!N \,\}. \tag{30}$$

In allen Fällen ist $R^* = \bigcup \{\, R^k \mid k \in I\!N \,\}$. Für (i) sei für $m, n, a \in M$.

$$(m,n) : a \; := \begin{cases} \{m\}, & \text{falls } n = a \\ \emptyset, & \text{sonst.} \end{cases}$$

Dann ist

$$\begin{aligned} R^* : A &= \bigcup \{ (m,n) : a \mid (m,n) \in R^*, a \in A \} \\ &= \bigcup \{ (m,n) : a \mid (m,n) \in R^k, a \in A, k \in I\!N \} \\ &= \bigcup \{ R^k : A \mid k \in I\!N \}. \end{aligned}$$

Für (ii) sei $(v_1, v_2) : w \;:=\; v_1 w v_2$ die Infixierung $\circ_1$ von Wörtern in Wortpaare. Damit ist

$$\begin{aligned} R^* : A &= \{ (v_1,v_2) : w \mid (v_1,v_2) \in R^*, w \in A \} \\ &= \{ (v_1,v_2) : w \mid (v_1,v_2) \in R^k, w \in A, k \in I\!N \} \\ &= \bigcup \{ R^k : A \mid k \in I\!N \}. \end{aligned}$$

Damit ist (27) gezeigt. Analog geht es für die Operationen $\circ_2, \circ_3$ in (iii). □

Das folgende Beispiel zeigt, daß $r^* : a$ nicht in jedem Halbmodul die kleinste Lösung von $x = r : x + a$ zu sein braucht.

Beispiel 5.22 Sei $\mathcal{K}' = (2^{\Sigma^*}, \cup, \emptyset, \cdot', \{\varepsilon\}, {}^*)$ aus $\mathcal{K} = (2^{\Sigma^*}, \cup, \emptyset, \cdot, \{\varepsilon\}, {}^*)$ durch die Änderung $A \cdot' B := B \cdot A$ entstanden. Das kartesische Produkt $\mathcal{K} \times \mathcal{K}'$ dieser Algebren, mit den komponentenweise definierten Operationen, ist nach Aufgabe 3.3 eine Kleene'sche Algebra. Für $A, B, C \subseteq \Sigma^*$ sei

$$(A \times B) : C \;:=\; \{ u \cdot v \cdot w \mid (u,v) \in A \times B,\, w \in C \}.$$

Damit ist $(\mathcal{K}, :)$ ist ein Halbmodul über $\mathcal{K} \times \mathcal{K}'$, da die Gleichungsaxiome zu : gelten, z.B.

$$\begin{aligned} ((A_1 \times A_2) \cdot (B_1 \times B_2)) : C &= ((A_1 \cdot B_1) \times (A_2 \cdot' B_2)) : C \\ &= (A_1 B_1 \times B_2 A_2) : C \\ &= A_1 B_1 C B_2 A_2 \\ &= (A_1 \times A_2) : ((B_1 \times B_2) : C). \end{aligned}$$

Wäre $(\mathcal{K}, :)$ ein Kleene'scher Halbmodul, so wäre $(A \times B)^* : C$ nach 5.19 die kleinste Lösung von $x = (A \times B) : x + C$. Das ist aber für $A = \{a\}, B = \{b\}, C = \{c\}$ mit verschiedenen $a, b, c \in \Sigma$ nicht der Fall: die Sprache

$$X := \bigcup \{ (A \times B)^n : C \mid n \in I\!N \} = \bigcup \{ A^n C B^n \mid n \in I\!N \} = \{ a^n c b^n \mid n \in I\!N \}$$

ist eine echte Teilmenge von

$$(A \times B)^* : C = (A^* \times B^*) : C = A^* C B^*,$$

aber X ist ebenfalls eine Lösung von $x = (A \times B) : x + C$ im Halbmodul $\mathcal{K}$.

Zwischen Matrizen über $\mathcal{K}$ und Vektoren über einem $\mathcal{K}$-Modul $\mathcal{A}$ kann man in naheliegender Weise eine Multiplikation erklären, die wieder zu einem Modul führt:

Satz 5.23 Sei $(\mathcal{A}, :)$ ein Halbmodul über der Kleene'schen Algebra $\mathcal{K}$.

(i) Dann ist $\mathcal{M}_{n,1}(\mathcal{A})$ ein Halbmodul über $\mathcal{M}_{n,n}(\mathcal{K})$, mit der Skalarmultiplikation

$$\begin{pmatrix} r_{1,1} & \cdots & r_{1,n} \\ \vdots & \ddots & \vdots \\ r_{1,1} & \cdots & r_{1,n} \end{pmatrix} : \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} \Sigma_{j=1}^n r_{1,j} : a_j \\ \vdots \\ \Sigma_{j=1}^n r_{n,j} : a_j \end{pmatrix}.$$

(ii) Ist $\mathcal{A}$ ein Kleene'scher Halbmodul über $\mathcal{K}$, so ist $\mathcal{M}_{n,1}(\mathcal{A})$ ein Kleene'scher Halbmodul über $\mathcal{M}_{n,n}(\mathcal{K})$.

Beweis: (i) Da die Addition auf $\mathcal{M}_{n,1}(\mathcal{A})$ komponentenweise erfolgt, ist mit $\mathcal{A}$ offenbar auch $\mathcal{M}_{n,1}(\mathcal{A})$ eine idempotente kommutative Halbgruppe. Die geforderten Eigenschaften der Skalarmultiplikation folgen aus denen der Skalarmultiplikation auf $\mathcal{A}$, etwa

$$\begin{aligned}
(r \cdot s) : a &= \left(\begin{pmatrix} r_{1,1} & \cdots & r_{1,n} \\ \vdots & \ddots & \vdots \\ r_{1,1} & \cdots & r_{1,n} \end{pmatrix} \cdot \begin{pmatrix} s_{1,1} & \cdots & s_{1,n} \\ \vdots & \ddots & \vdots \\ s_{1,1} & \cdots & s_{1,n} \end{pmatrix} \right) : \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix} \\
&= \begin{pmatrix} & \vdots & \\ \cdots & \Sigma_{k=1}^n r_{i,k} \cdot s_{k,j} & \cdots \\ & \vdots & \end{pmatrix}_{i,j} : \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix} \\
&= \begin{pmatrix} \Sigma_{j=1}^n (\Sigma_{k=1}^n r_{1,k} \cdot s_{k,j}) : a_j \\ \vdots \\ \Sigma_{j=1}^n (\Sigma_{k=1}^n r_{n,k} \cdot s_{k,j}) : a_j \end{pmatrix} = \begin{pmatrix} \Sigma_{j=1}^n \Sigma_{k=1}^n \ r_{1,k} : (s_{k,j} : a_j) \\ \vdots \\ \Sigma_{j=1}^n \Sigma_{k=1}^n \ r_{n,k} : (s_{k,j} : a_j) \end{pmatrix} \\
&= \begin{pmatrix} r_{1,1} & \cdots & r_{1,n} \\ \vdots & \ddots & \vdots \\ r_{1,1} & \cdots & r_{1,n} \end{pmatrix} : \begin{pmatrix} \Sigma_{j=1}^n \ s_{k,j} : a_j \\ \vdots \\ \Sigma_{j=1}^n \ s_{k,j} : a_j \end{pmatrix} = r : (s : a).
\end{aligned}$$

(ii) Sei nun $\mathcal{A}$ ein Kleene'scher Halbmodul über $\mathcal{K}$, und $R : X + E \leq X$ in $\mathcal{M}_{n,1}(\mathcal{A})$ mit der duch $+$ gegebenen partiellen Ordnung $\leq$ auf $\mathcal{M}_{n,1}(\mathcal{A})$. Es genügt zu zeigen, daß $R^* : X \leq X$ ist: dann folgt aus $R : X + E \leq X$ einerseits $R : X \leq X$, also $R^* : X \leq X$, und andererseits $E \leq X$, also mit Monotonie auch $R^* : E \leq R^* : X \leq X$. Der Beweis von $R^* : X \leq X$ geht genau wie in Lemma 5.4. □

Aus 5.23 und 5.19 folgt, daß lineare Gleichungssysteme über $(\mathcal{R}, \mathcal{A}, :)$ kleinste Lösungen bezüglich der durch $a \leq b :\iff a + b = b$ definierten Ordnung haben:

Korollar 5.24 Sei $(\mathcal{A}, :)$ ein Kleene'scher Halbmodul über $\mathcal{K}$. Jede Matrixgleichung

$$X = R : X + B, \qquad \text{mit } R \in \mathcal{M}_{n,n}(\mathcal{K}) \text{ und } B \in \mathcal{M}_{n,1}(\mathcal{A}),$$

hat eine kleinste Lösung, nämlich $R^* : B \in \mathcal{M}_{n,1}(\mathcal{A})$.

In einem Halbmodul $\mathcal{M} = (\mathcal{R}, \mathcal{A}, :)$ definiert jedes $r \in R$ durch $f_r(a) := r : a$ eine Abbildung $f_r : A \to A$, die wegen der Distributivitätsforderung an : ADDITIV ist, d.h. $f_r(a_1 + a_2) = f_r(a_1) + f_r(a_2)$ erfüllt. Das setzt sich natürlich auf die Matrizen fort:

Proposition 5.25 Ist $(S, +, 0, :)$ ein Halbmodul über K, so definiert jede Matrix $A \in \mathcal{M}_{m,n}(\mathcal{K})$ durch

$$f_A(x) = A : x = \begin{pmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \cdots & a_{m,n} \end{pmatrix} : \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^n a_{1,j} : x_j \\ \vdots \\ \sum_{j=1}^n a_{m,j} : x_j \end{pmatrix}$$

eine additive Abbildung $f_A : S^{(n)} \to S^{(m)}$, d.h. es ist $f_A(x + y) = f_A(x) + f_A(y)$ für alle $x, y \in S^{(n)}$.

Bemerkung 5.26 Wenn man die Sprachverkettung und -Iteration ignoriert, ist $\mathcal{L}_\Sigma$ nach 5.20 ein Linksmodul über $\mathcal{R} = \mathcal{L}_{\Sigma,\Sigma}$. Die von einer Relation $R \subseteq \Sigma^* \times \Sigma^*$ repräsentierte Funktion $f_R : \mathcal{R} \to \mathcal{R}$ ist zwar additiv, aber i.a. nicht rechtslinear, da f_R das Sprachprodukt normalerweise nicht respektiert: ist $R = \{(b_1, b_2)\}$ und $A_i = \{a_i\}$ mit $b_1 a_1 a_2 b_2 \neq b_1 a_1 b_2 a_2$, so ist

$$R : (A_1 \cdot A_2) \neq (R : A_1) \cdot A_2.$$

Wie wir zu einer Kleene'schen Algebra $\mathcal{K}$ und $A \subseteq K$ in Kapitel 2 die reguläre Unteralgebra $Reg_A^{\mathcal{K}}$ definiert hatten, definieren wir nun spezielle Untermoduln:

Definition 5.27 Sei $\mathcal{S} = (S, +, 0, :)$ ein Kleene'scher Halbmodul über $\mathcal{K}$. Der VON $A \subseteq S$ ERZEUGTE UNTERMODUL von $\mathcal{S}$ ist

$$Lin_A^{\mathcal{S}} := \bigcap \{\, M \mid A \subseteq M \subseteq S,\ M + M \subseteq M,\ K : M \subseteq M \,\}$$

zusammen mit den Einschränkungen der Operationen aus $\mathcal{S}$ auf $Lin_A^{\mathcal{S}}$.

Die Bezeichnung rührt daher, daß $Lin_A^{\mathcal{S}}$ aus den LINEARKOMBINATIONEN von Elementen aus A (mit Koeffizienten aus K) besteht, d.h. aus allen Summen

$$\Sigma_{i=1}^m\, r_i : a_i \qquad \text{mit } m \in I\!N,\ r_1, \ldots, r_m \in K,\ a_1, \ldots, a_m \in A.$$

Das ist klar, da $Lin_A^{\mathcal{S}}$ die kleinste Obermenge von A in S ist, die unter Addition und Multiplikation mit Skalaren aus $\mathcal{K}$ abgeschlossen ist.

Definition 5.28 Sei $\mathcal{M} = (Reg^{\mathcal{L}}_{\Sigma,\Sigma}, \mathcal{A}, :)$ der Kleene'sche Modul $\mathcal{A} = (2^{\Sigma^*}, \cup, \emptyset)$ über der Algebra der regulären Wortrelationen $Reg_{\Sigma,\Sigma} \subseteq \mathcal{L}_{\Sigma,\Sigma}$ aus Beispiel 3.11, wobei

$$R : B := \{\, a_1 b a_2 \mid (a_1, a_2) \in R, b \in B \,\}$$

die punktweise Einfügung einer Wortmenge B in eine Wortrelation R ist. Den von $\Sigma \cup \{\varepsilon\}$ erzeugten Untermodul von $\mathcal{L} = (\mathcal{A}, :)$ bezeichnen wir mit

$$Lin^{\mathcal{L}}_{\Sigma\cup\{\varepsilon\}} = (Lin_\Sigma, \cup, \emptyset, :, \{\varepsilon\}, \{a\})_{a\in\Sigma}.$$

Sprachen in $Lin^{\mathcal{L}}_{\Sigma\cup\{\varepsilon\}}$ sind also genau diejenigen, die sich durch Vereinigung und Einfügung in reguläre Wortrelationen aus den Sprachen $\emptyset$, $\{\varepsilon\}$ und $\{a\}$, mit $a \in \Sigma$, erzeugen lassen.

5.4 Lineare Sprachen und lineare Ausdrücke

Wir wenden nun die Begriffe des vorigen Abschnitts auf formale Sprachen über einem Alphabet Σ an.

Definition 5.29 Ein LINEARES GLEICHUNGSSYSTEM (oder eine LINEARE GRAMMATIK) über Σ ist ein Gleichungssystem

$$\begin{array}{rcl} x_1 & = & r_1(x_1, \ldots, x_m) \\ \vdots & \vdots & \vdots \\ x_m & = & r_m(x_1, \ldots, x_m) \end{array} \tag{31}$$

dessen rechte Seiten r_i Summen von Wörtern w oder uxv sind, wobei $u, v, w \in \Sigma^+ \cup \{0, 1\}$ und x eine der REKURSIONSVARIABLEN $x_1, \ldots, x_m$ von (31) ist. Eine Sprache $L \subseteq \Sigma^*$ heißt LINEAR, wenn sie eine Komponente der kleinsten Lösung eines linearen Gleichungssystems über Σ ist. (Die Existenz der kleinsten Lösung wird in Satz 5.33 gezeigt.)

Im Unterschied zu rechtslinearen Grammatiken, wo die einzige erlaubte syntaktische Konstruktionsmöglichkeit die explizite Angabe eines Ausdrucks $w \in \Sigma^*$ oder das Anfügen eines Präfixes $u \in \Sigma^*$ an einen Ausdruck einer Kategorie x war, ist bei linearen Grammatiken das Anhängen von Präfixen und Suffixen erlaubt. Das braucht man etwa zur Beschreibung der Wortarten natürlicher Sprachen:

Beispiel 5.30 Die Wortklassen des Deutschen wie Adjektive, Nomina und Verben, kurz A, N, V, sind simultan definiert und sowohl unter gewissen Präfixen als auch Suffixen abgeschlossen (in starker Idealisierung):

$$\begin{array}{rclclclcl} A & = & \mathsf{un}A & + & N\,\mathsf{lich} & + & V\,\mathsf{bar} & + & \ldots \\ N & = & A\,\mathsf{keit} & + & \mathsf{un}N & + & V\,\mathsf{ung} & + & \ldots \\ V & = & A\,\mathsf{en} & + & N\,\mathsf{en} & + & \mathsf{ver}V & + & \ldots \end{array}$$

Darunter fallen Adjektive wie unschön, unlösbar, handlich, machbar, Nomina wie Eitelkeit, Machbarkeit, Untat, Handlung, oder Verben wie schönen, färben, verfärben. Als syntaktische Strukturen erhält man Bäume, bei denen die Rekursionsvariablen alle auf einem Pfad liegen:

```
        N
       / \
      A   keit
     / \
   Un   A
       / \
      V   bar
     / \
  zer   V
        |
       leg
```

Falls Präfix- und Suffixbildung bei der Wortbildung beliebig tief geschachtelt auftreten darf –man denke an die angebliche "Unkaputtmachbarkeit" der deutschen Sprache– , kann man nach der folgenden Proposition schon die Wortbildung natürlicher Sprachen nicht mehr mit rechtslinearen Grammatiken beschreiben.

Proposition 5.31 Die regulären Sprachen über Σ bilden eine echte Teilklasse der linearen Sprachen über Σ.

Beweis: Nach Proposition 4.8 können reguläre Sprachen durch rechtslineare, also auch durch lineare Gleichungssysteme definiert werden. Daß die Umkehrung nicht gilt, zeigt die nach 4.21 nicht reguläre Sprache

$$L = \{ a^n b^n \mid n \geq 0 \}$$

die offenbar die kleinste Lösung der linearen Gleichung $x = axb + 1$ ist. $\square$

Wir stellen nun den Zusammenhang zwischen dem im vorigen Abschnitt definierten Modul $Lin^{\mathcal{L}}_{\Sigma}$ über $Reg^{\mathcal{L}}_{\Sigma,\Sigma}$ und den linearen Grammatiken her. Einen Summanden uxv in einer linearen Grammatik kann man als die Einfügung $(u, v) : x$ von x in den Kontext (u, v) verstehen.

Ein BIREGULÄRER AUSDRUCK ÜBER Σ ist ein regulärer Ausdruck über

$$\Sigma' := \{ (a, \varepsilon) \mid a \in \Sigma \} \cup \{ (\varepsilon, a) \mid a \in \Sigma \}.$$

Bireguläre Ausdrücke p über Σ interpretieren wir durch reguläre Wortrelationen $L(p) = P \in Reg^{\mathcal{L}}_{\Sigma,\Sigma}$, wobei

$$L((\varepsilon, a)) = \{(\varepsilon, a)\}, \quad L((a, \varepsilon)) = \{(a, \varepsilon)\} \quad \text{und} \quad L(1) = \{(\varepsilon, \varepsilon)\}$$

für die atomaren Relationen benutzt wird und die (iterierte) Einfügung einer Wortrelation in eine andere als (iteriertes) Produkt.

Elemente von $Lin^{\mathcal{L}}_{\Sigma}$ können wir dann mit folgenden Ausdrücken benennen:

Definition 5.32 Die LINEAREN AUSDRÜCKE ÜBER Σ sind definiert durch:

(i) $0, 1$ und jedes $a \in \Sigma$ ist ein linearer Ausdruck über Σ.

(ii) Sind r, s lineare Ausdrücke über Σ und ist p ein biregulärer Ausdruck über Σ, so sind $(r + s)$ und $p : r$ lineare Ausdrücke über Σ.

Lineare Ausdrücke über Σ werden dann durch Sprachen $\mathcal{L}(r)$ im Modul aus Beispiel 5.20 interpretiert.

Satz 5.33 (Gruska, 1971) Für $L \subseteq \Sigma^*$ sind folgende Aussagen äquivalent:

(i) $L \in Lin^{\mathcal{L}}_{\Sigma\cup\{\varepsilon\}}$, d.h. L ist eine Linearkombination endlicher Sprachen mit regulären Wortrelationen über Σ,

(ii) $L = L(p)$ für einen linearen Ausdruck p über Σ,

(iii) L ist Komponente der kleinsten Lösung eines linearen Gleichungssystems über Σ.

Beweis: Die Äquivalenz von (i) und (ii) folgt direkt aus den Definitionen.

(ii) $\Rightarrow$ (iii) Durch Induktion über den Aufbau linearer Ausdrücke r über Σ.

$r \in \{0, 1\} \cup \Sigma$: Dann ist $L(r)$ die kleinste Lösung von $x_0 = r$.

$r \equiv (r_1 + r_2)$: Nach Induktion sind $L(r_1)$ und $L(r_2)$ die ersten Komponenten der kleinsten Lösungen zweier linerarer Gleichungssysteme

$$\begin{array}{rcl} x_1 & = & r_1(x_1, \ldots, x_m) \\ \vdots & \vdots & \vdots \\ x_m & = & r_m(x_1, \ldots, x_m) \end{array} \quad \text{und} \quad \begin{array}{rcl} x_{m+1} & = & r_{m+1}(x_{m+1}, \ldots, x_{m+k}) \\ \vdots & \vdots & \vdots \\ x_{m+k} & = & r_{m+k}(x_{m+1}, \ldots, x_{m+k}). \end{array}$$

Zu diesen fügen wir die lineare Gleichung $x_0 = x_1 + x_{m+1}$ hinzu. Offenbar ist $L(r)$ die x_0-Komponente seiner kleinsten Lösung.

$r \equiv p : s$: Nach Induktion ist $L(s)$ die erste Komponente der kleinsten Lösung eines linearen Gleichungssystems

$$\begin{array}{rcl} x_1 & = & r_1(x_1, \ldots, x_m) \\ \vdots & \vdots & \vdots \\ x_m & = & r_m(x_1, \ldots, x_m). \end{array}$$

Wir zeigen durch Induktion über p, daß $L(p : x_1)$ durch ein Gleichungssystem

$$\begin{array}{rcl} x_{m+1} & = & r_{m+1}(x_1, x_{m+1}, \ldots, x_{m+k}) \\ \vdots & \vdots & \vdots \\ x_{m+k} & = & r_{m+k}(x_1, x_{m+1}, \ldots, x_{m+k}) \end{array}$$

über $\Sigma \cup \{x_1\}$ definierbar ist, das in $x_{m+1}, \ldots, x_{m+k}$ *und* x_1 linear ist: in einem Summanden dieses Systems kommt x_1 höchstens einmal vor, und auch das nur dann, wenn $x_{m+1}, \ldots, x_{m+k}$ nicht darin vorkommen. Daher ist die Zusammenfassung der beiden Systeme ein ein lineares Gleichungssystem, und $L(p : s)$ ist die x_{m+1}-Komponente seiner kleinsten Lösung.

Wir beweisen die Aussage über $L(p : x)$ entsprechend dem Aufbau von p:

(i) $p \in \{0, 1\} \cup \{\, (a, \varepsilon), (\varepsilon, a) \mid a, b \in \Sigma \,\}$: Beispielsweise wird $L((a, \varepsilon) : x)$ durch die lineare Gleichung $x_0 = ax$ definiert, in der x nur wie angegeben auftritt.

(ii) $p \equiv (p_1 + p_2)$: Es ist $L((p_1 + p_2) : x) = L(p_1 : x) \cup L(p_2 : x)$. Nach Induktion ist $L(p_1 : x)$ durch ein in x und Rekursionsvariablen $x_1, \ldots, x_m$ und $L(p_2 : x)$ durch ein in x und Rekursionsvariablen $x_{m+1}, \ldots, x_{m+k}$ lineares Gleichungssystem definierbar. Mit $x_0 = x_1 + x_{m+1}$ erweitert, erhalten wir ein in x und $x_0, \ldots, x_{m+k}$ lineares Gleichungssystem, das $L(p : x)$ definiert.

(iii) $p \equiv (p_1 \cdot p_2)$: Es ist $L((p_1 \cdot p_2) : x) = L(p_1) : L(p_2 : x)$. Induktiv ist $L(p_2 : x)$ durch ein in x und Rekursionsvariablen $x_1, \ldots, x_m$ lineares Gleichungssystem definiert, und $L(p_1 : x_1)$ durch ein in x_1 und Rekursionsvariablen $x_{m+1}, \ldots, x_{m+k}$ lineares System. Das daraus zusammengesetzte Gleichungssystem ist linear in x und seinen Rekursionsvariablen $x_1, \ldots, x_{m+k}$ und definiert $L(p : x)$ durch seine x_{m+1}-Komponente.

(iv) $p \equiv q^*$: Für eine zweistellige Wortrelation Q und eine Wortmenge S ist die Menge $Q^* : S$ die kleinste Lösung von $x_0 = Q : x_0 + S$. Nach Induktion ist $L(q : x_0)$ durch ein in x_0 und Rekursionsvariablen $x_1, \ldots, x_m$ lineares Gleichungssystem über $\Sigma \cup \{x_0\}$ definierbar. Erweitern wir dieses System um $x_0 = x_1 + x$ (an Stelle von $x_0 = q : x_0 + x$), so erhalten wir ein in x und den Rekursionsvariablen $x_0, \ldots, x_m$ lineares System, das $L(p : x)$ definiert.

$\Leftarrow$: Sei $L = L(x_0)$ für die kleinste Lösung $L(x_0), \ldots, L(x_m)$ des linearen Gleichungssystems

$$\begin{array}{rcl} x_0 & = & r_0(x_0, \ldots, x_m) \\ \vdots & \vdots & \vdots \\ x_m & = & r_m(x_0, \ldots, x_m) \end{array}$$

über Σ. Für die Induktion erlauben wir allgemeiner, daß $r_i(x_0, \ldots, x_m)$ ein *linearer* Ausdruck über $\Sigma \cup \{x_0, \ldots, x_m\}$ ist, wobei die Variablen x_i nur in Summanden der Form $p : x_i$ mit biregulärem Ausdruck p über Σ – also ohne Variable – vorkommen. Für das vorgegebenen System ist das der Fall, da man für $u = a_1 \cdots a_n \in \Sigma^*$ und $v = b_1 \cdots b_k$ die Summanden vx_iw durch

$$p : x_i := ((a_1, \varepsilon) \cdots (a_n, \varepsilon) \cdot (\varepsilon, b_k) \cdots (\varepsilon, b_1)) : x_i$$

darstellen kann. Ein lineares Gleichungssystem über Σ kann man also als System

$$\begin{array}{rcl} x_1 & = & q_{1,1} : x_1 + \ldots + q_{1,m} : x_m + b_1 \\ \vdots & \vdots & \vdots \\ x_m & = & q_{m,1} : x_1 + \ldots + q_{m,m} : x_m + b_m \end{array}$$

bzw. als Matrixgleichung $X = Q : X + B$ im Kleene'schen Halbmodul $\mathcal{L}_\Sigma$ über der Algebra $Reg^{\mathcal{L}}_{\Sigma,\Sigma}$ der regulären Wortrelationen schreiben. Diese Gleichung hat nach 5.24 eine kleinste Lösung, nämlich

$$\mu X(Q : X + B) = Q^* : B.$$

Die Einträge in Q^* sind nach der Rekursionsformel für iterierte Matrizen durch reguläre Ausdrücke über den Einträgen aus Q gegeben, also reguläre Wortrelationen. Daher sind die Komponenten von $Q^* : B$ durch Einfügen endlicher Sprachen in reguläre Wortrelationen entstanden, also linear. $\square$

Beispiel 5.34 Betrachte das lineare Gleichungssystem

$$\begin{array}{rcccccccc} x & = & axb & & & + & ay & & \\ y & = & xc & + & ya & + & by & + & 1 \end{array}$$

Im Modul $\mathcal{L}_\Sigma$ über $Reg^{\mathcal{L}}_{\Sigma,\Sigma}$, mit $\Sigma = \{a, b, c\}$, läßt es sich in der Form

$$\begin{array}{rcccccc} x & = & (a,b) : x & + & (a,1) : y & & \\ y & = & (1,c) : x & + & ((1,a) + (b,1)) : y & + & 1 \end{array}$$

schreiben, oder als Matrixgleichung

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} (a,b) & (a,1) \\ (1,c) & (1,a) + (b,1) \end{pmatrix} : \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Da $\mathcal{L}_\Sigma$ mit der Infixierung in Wortpaare nach 5.21 ein Kleene'scher Modul über $Reg^{\mathcal{L}}_{\Sigma,\Sigma}$ ist, hat diese Matrixgleichung nach 5.24 eine kleinste Lösung, nämlich

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} (a,b) & (a,1) \\ (1,c) & (1,a)+(b,1) \end{pmatrix}^* : \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Nach der Rekursionsformel

$$E^* = \begin{pmatrix} A & B \\ C & D \end{pmatrix}^* := \begin{pmatrix} F^* & F^*BD^* \\ D^*CF^* & D^* + D^*CF^*BD^* \end{pmatrix}, \qquad \text{mit } F := A + BD^*C,$$

ist dies

$$\begin{aligned} x &= F^* : 0 + F^*BD^* : 1 \\ &= (A + BD^*C)^*BD^* : 1 \\ &= ((a,b) + (a,1)((1,a)+(b,1))^*(1,c))^*(a,1)((1,a)+(b,1))^* : 1 \end{aligned}$$

Bemerkung: Erlaubt man zusätzlich die Iteration für Wörter, so kann man $((1,a)+(b,1))^*$ durch (a^*,b^*) ersetzen und kommt auf

$$x = ((a,b) + (ab^*, ca^*))^*(ab^*, a^*) : 1 = ((a,b) + (ab^*, ca^*))^* : ab^*a^*.$$

Proposition 5.35 Eine Sprache $L \subseteq \Sigma^*$ ist genau dann linear, wenn sie der Kollaps $\{\, uv \mid (u,v) \in P \,\}$ einer regulären Wortrelation $P \subseteq \Sigma^* \times \Sigma^*$ ist.

Beweis: Es genügt zu zeigen, daß es zu jedem linearen Ausdruck r über Σ einen biregulären Ausdruck p über Σ mit $L(r) = L(p : 1)$ gibt. Das sieht man direkt durch Induktion über den Aufbau der linearen Ausdrücke. □

Automaten für lineare Sprachen

Wie bei rechtslinearen Grammatiken genügt es, in linearen Gleichungssystemen nur Summanden einer einfachen Form zu erlauben. Durch eine kleine Erweiterung des Beweises von Satz 4.8 erhält man folgende Normalform für lineare Grammatiken:

Proposition 5.36 Jede lineare Sprache ist durch ein lineares Gleichungssystem (31) definierbar, dessen Summanden von der Form 0, 1, ax, oder xa sind, mit $a \in \Sigma$ und $x \in \{x_1, \ldots, x_m\}$.

Ein lineares Gleichungssystem in dieser Normalform entspricht einem endlichen endlichen Automaten (über Σ' bzw. $Reg^{\mathcal{L}}_{\Sigma,\Sigma}$)

$$\mathcal{A} = (\{x_1, \ldots, x_m\}, \vdash_{(a,\varepsilon)}, \vdash_{(\varepsilon,a)}, x_1, F)_{a\in\Sigma},$$

zur Erkennung einer regulären Wortrelation R, wobei die Übergangsrelationen und die Endzustandsmenge durch

$$\begin{aligned} x_i \vdash_{(a,\varepsilon)} x_j \quad &:\Longleftrightarrow \quad ax_j \text{ ist Summand von } r_i, \\ x_i \vdash_{(\varepsilon,a)} x_j \quad &:\Longleftrightarrow \quad x_j a \text{ ist Summand von } r_i, \\ x_i \in F \quad &:\Longleftrightarrow \quad 1 \text{ ist Summand von } r_i \end{aligned}$$

definiert sind. Ein Eingabepaar (u, v) ist dabei als Produkt (im Sinne der Einfügung) von Buchstaben (a, ε) und (ε, a) zu verstehen, und die Grundoperationen $\vdash_{(a,\varepsilon)}$ und $\vdash_{(\varepsilon,b)}$ bedeuten Lesen *am Anfang der ersten Komponente* bzw. Lesen *am Ende der zweiten Komponente* von (u, v), da ja $(a, \varepsilon) \cdot (u', v') = (au, v')$ und $(\varepsilon, b) \cdot (u', v') = (u', v'b)$.

Da aber $(a, \varepsilon)(\varepsilon, b) = (\varepsilon, b)(a, \varepsilon)$ und daher die Zerlegung von (u, v) in ein Produkt von Buchstaben nicht eindeutig ist, sind solche Automaten nicht deterministisch: es muß stets zwischen Lesen am Anfang und Lesen am Ende gewählt werden.

Da nach Proposition 5.35 die linearen Sprachen L gerade die Kollabierungen $L = R : 1$ regulärer Wortrelationen R sind, können wir diese Automaten $\mathcal{A}$ auch als Automaten zur Erkennung linearer Sprachen verwenden.

Man kann auch die Definition von Automaten über Kleene'schen Algebren wie folgt zu Automaten über Kleene'schen Moduln $\mathcal{M} = (\mathcal{K}, \mathcal{A}, :)$ verallgemeinern:

Ein LINEARER AUTOMAT über dem Kleene'schen $\mathcal{K}$-Modul $(\mathcal{A}, :)$ ist ein Tripel $\langle u, B, v\rangle$, wobei

$$u \in M_{n,1}(\{0, 1\}),\ B \in M_{n,n}(\mathcal{K}),\ v \in M_{n,1}(\mathcal{A}).$$

Die VON $\langle u, B, v\rangle$ ERKANNTE SPRACHE ist

$$L(\langle u, B, v\rangle) := u^T : (B^* : v) \quad \in M_{1,1}(\mathcal{A}) = A.$$

$B^* : v$ liefert ein Tupel von Elementen im Modul $\mathcal{A}$, und die Multiplikation mit u^T bildet eine Summe von einigen dieser Elemente.

Falls $\mathcal{K} = \mathcal{L}_{\Sigma,\Sigma}$ und $\mathcal{A} = \mathcal{L}_\Sigma$ ist, können wir $v \in M_{n,1}(\{0, 1\})$ annehmen, da

$$L \mapsto \{\, (w, 1) \mid w \in L \,\}$$

eine Einbettung von $\mathcal{L}_\Sigma$ in $\mathcal{L}_{\Sigma,\Sigma}$ ist.

Abschluß- und Entscheidbarkeitseigenschaften

Eine m-LINEARE SPRACHE $L \subseteq \Sigma^*$ ist eine endliche Vereinigung von Sprachen der Form $L_1 \cdots L_m$, worin jedes L_i eine lineare Sprache ist. Mit $m\text{-}Lin_\Sigma$ bezeichnen wir die Menge der m-linearen Sprachen über Σ.

Die Klasse der linearen Sprachen hat u.a. folgende Abschlußeigenschaften:

Satz 5.37 Sei Lin_Σ die Menge der durch ein lineares Gleichungssystem definierbaren Sprachen über Σ. Dann gilt:

(i) $Reg_\Sigma \subset Lin_\Sigma = 1\text{-}Lin_\Sigma \subset \ldots\ m\text{-}Lin_\Sigma \subset (m+1)\text{-}Lin_\Sigma \ldots$.

(ii) Lin_Σ ist abgeschlossen unter Vereinigung.

(iii) Lin_Σ ist nicht abgeschlossen unter Verkettung, Iteration, oder Substitution von Buchstaben durch lineare Sprachen über Σ.

Beweis: (Skizze) (i) Die erste Inklusion wurde in Proposition 5.31 gezeigt. Für die übrigen Inklusionen braucht man ein Wiederholungslemma (vgl. Aufgabe 5.5). Damit zeigt man, daß L^{m+1} mit $L = \{\, a^n b^n \mid n \in I\!N \,\}$ aus 5.31 in $(m+1)\text{-}Lin_\Sigma$ liegt, aber nicht in $m\text{-}Lin_\Sigma$.

(ii) Das folgt aus 5.33 mit der Summe zweier linearer Ausdrücke.

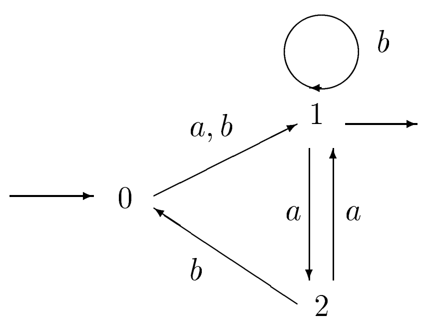
(iii) Da LL mit L aus dem Beweis von 5.31 aus der endlichen, also linearen Sprache $\{aa\}$ durch die Ersetzung $[L/a]$ hervorgeht, ist Lin_Σ nicht unter Substitution abgeschlossen; ebenso nicht unter Verkettung. Mit einem Wiederholungslemma sieht man auch, daß L^* nicht linear ist. □

Entscheidbarkeitseigenschaften der linearen Sprachen holen wir in Abschnitt 8.5 nach. Im Unterschied zum regulären Fall gibt es kein Verfahren, die Äquivalenz linearer Ausdrücke, d.h. ob $L(r) = L(s)$ gilt, festzustellen. Insbesondere kann das Äquivalenzproblem nicht auf eine Isomorphie von minimalen linearen Automaten zurückgeführt werden.

Aufgaben

Aufgabe 5.1 Man konstruiere zum endlichen Automaten

$\mathcal{A} :=$



einen regulären Ausdruck r mit $L(r) = L(\mathcal{A})$ durch Aufstellen der Matrixdarstellung $\mathcal{A} = \langle v_i, M, v_o\rangle$ und Berechnen von

$$L(\langle v_i, M, v_o\rangle) = v_i^T \cdot M^* \cdot v_o$$

Welche Felder von M^* muß man dafür nur berechnen?

Aufgabe 5.2 Seien $L(x), L(y) \subseteq \Sigma^* = \{a, b\}^*$ die Komponenten der kleinsten Lösung des linearen Gleichungssystems (der linearen kontextfreien Grammatik)

$$\begin{aligned} x &= axbb + bxa + bbya + c \\ y &= ax + yc + b. \end{aligned}$$

Man gebe einen variablenfreien linearen Ausdruck s mit $L(x) = L(s)$ an.

Aufgabe 5.3 Zeige, daß die Komponenten der kleinsten Lösung eines *links*linearen Gleichungssystems

$$\begin{array}{ccc} x_1 & = & r_1(x_1, \ldots, x_n) \\ \vdots & \vdots & \vdots \\ x_n & = & r_n(x_1, \ldots, x_n) \end{array},$$

d.h. wenn die r_i Summen von Ausdrücken $x_j w$ oder w mit $w \in \Sigma^*$ sind, reguläre Sprachen sind.

Aufgabe 5.4 (i) Formuliere und beweise ein Wiederholungslemma für die Klasse der linearen Sprachen. (Benutze die Automaten für lineare Sprachen und modifiziere die Idee aus dem Beweis von Lemma 4.20.)

(ii) Zeige mit diesem Wiederholungslemma, daß $L = \{ a^n b^n a^m b^m \mid n \in \mathbb{N}, m \in \mathbb{N} \}$ keine lineare Sprache über $\{a, b\}$ ist, falls $a \neq b$.

Aufgabe 5.5 Zeige, daß ein Halbmodul $(\mathcal{A}, :)$ über der Kleene'schen Algebra $\mathcal{R}$ genau dann ein Kleene'scher Halbmodul ist, also

$$\forall r \in R\ \forall a, b \in A\ (r : b + a \leq b\ \rightarrow\ r^* : a \leq b) \tag{32}$$

erfüllt, wenn er

$$\forall r \in R\ \forall b \in A\ (r : b \leq b\ \rightarrow\ r^* : b \leq b) \tag{33}$$

erfüllt.

6 Gleichungsdefinierbare und kontextfreie Sprachen

Die bisher betrachteten Sprachdefinitionen durch reguläre Ausdrücke erlauben es noch nicht, nützliche Sprachen wie die in der Mathematik üblichen Formelsprachen zu definieren. Die Definition solcher Sprachen erfolgt INDUKTIV, wobei zur Einführung der Ausdrücke einer syntaktischen Kategorie schon andere Ausdrücke der zu definierenden Kategorie verwendet werden. Die frühere Definition 2.1 der regulären Ausdrücke ist eine solche induktive Definition. Oft werden mehrere syntaktische Kategorien simultan induktiv definiert, wie etwa die Terme und Formeln der Prädikatenlogik:

Beispiel 6.1 Die TERME und FORMELN DER PRÄDIKATENLOGIK definiert man –relativ zu vorgegebenen Listen V von Variablen, $\mathcal{R}$ von Relations- und $\mathcal{F}$ von (mit einer Stelligkeit $n \in I\!N$ versehenen) Funktionszeichen– wie folgt:

(i) Ist $x \in V$, so ist x ein Term.

(ii) Ist $f \in \mathcal{F}$ n-stellig und sind $t_1, \ldots, t_n$ Terme, so ist auch $f(t_1, \ldots, t_n)$ ein Term.

(iii) Sind t_1 und t_2 Terme, so ist $t_1 = t_2$ eine Formel.

(iv) Ist $R \in \mathcal{R}$ n-stellig und sind $t_1, \ldots, t_n$ Terme, so ist $R(t_1, \ldots, t_n)$ eine Formel.

(v) Ist φ eine Formel, so ist auch $\neg\varphi$ eine Formel.

(vi) Sind φ_1 und φ_2 Formeln, so sind auch $(\varphi_1 \wedge \varphi_2)$, $(\varphi_1 \vee \varphi_2)$, $(\varphi_1 \to \varphi_2)$ und $(\varphi_1 \leftrightarrow \varphi_2)$ Formeln.

(vii) Ist φ eine Formel und x eine Variable, so sind auch $\exists x\, \varphi$ und $\forall x\, \varphi$ Formeln.

(viii) Nur, was mit endlich vielen Anwendungen von i) bis vii) erzeugt werden kann, ist eine Formel.

Bedingung viii) bedeutet, daß die Menge der Terme und Formeln die *kleinsten* Mengen von Zeichenreihen sind, die den Bedingungen in i) bis vii) genügen.

6.1 Reguläre Gleichungssysteme und μ-reguläre Ausdrücke

Diese induktive Art der Definition formulieren wir jetzt als eine Erweiterung der Definition durch reguläre Ausdrücke. Dazu erlauben wir in regulären Ausdrücken über Σ in Zukunft auch Variable x_i, die durch die Schreibweise $r(x_1, \ldots, x_n)$ hervorgehoben werden.

Definition 6.2 Sei Σ ein endliches Alphabet und X eine unendliche Liste von Variablen, mit $\Sigma \cap X = \emptyset$. Ein REGULÄRES GLEICHUNGSSYSTEM ÜBER Σ ist eine Folge S von Gleichungen,

$$\begin{array}{ccc} x_1 & = & r_1(x_1, \ldots, x_m) \\ \vdots & \vdots & \vdots \\ x_m & = & r_m(x_1, \ldots, x_m), \end{array}$$

in der jedes $r_i(x_1,\dots,x_m)$ ein regulärer Ausdruck über Σ ist, dessen Variable unter den paarweise verschiedenen REKURSIONSVARIABLEN $x_1,\dots,x_m \in X$ vorkommen.

Wir suchen Lösungen für solche Gleichungssysteme sowohl in der Algebra $\mathcal{L}_\Sigma$ aller Sprachen über Σ als auch in Relationenalgebren $\mathcal{R} = (2^{M\times M}, \cup, 0, \cdot, 1, {}^*, \vdash_a)_{a\in\Sigma}$. Wie man die Lösung findet, zeigen wir allgemeiner für gewisse Kleene'sche Algebren $\mathcal{K}$ mit ausgewählten $a_i^{\mathcal{K}} \in K$.

Definition 6.3 Eine Kleene'sche Algebra $\mathcal{K} = (K, +, 0, \cdot, 1, {}^*)$ heißt STETIG, falls gilt:

(i) Die durch $a \le b :\iff a+b=b$ definierte partielle Ordnung ist vollständig, d.h. für jede gerichtete Teilmenge $X \subseteq K$ gibt es eine kleinste obere Schranke, $\bigsqcup X \in K$. Dabei heißt $X \subseteq K$ GERICHTET, falls

$$\forall x_1 \in X\, \forall x_2 \in X\, \exists x \in X\ (x_1 \le x\ \wedge\ x_2 \le x).$$

Die kleinste obere Schranke, $\bigsqcup X$, ist das (!) $y \in K$ mit der Eigenschaft

$$\forall x \in X\, x \le y\ \wedge\ \forall z \in K\, (\forall x \in X\, x \le z\ \rightarrow\ y \le z).$$

(ii) Die Abbildungen $+, \cdot$ und * sind stetig, d.h. sie sind monoton und für alle gerichteten Teilmengen $X_1, X_2 \subseteq K$ gelten

$$\begin{aligned}(\bigsqcup X_1 + \bigsqcup X_2) &= \bigsqcup\{\, x_1 + x_2 \mid x_1 \in X_1,\, x_2 \in X_2 \,\},\\ (\bigsqcup X_1 \cdot \bigsqcup X_2) &= \bigsqcup\{\, x_1 \cdot x_2 \mid x_1 \in X_1,\, x_2 \in X_2 \,\},\\ (\bigsqcup X_1)^* &= \bigsqcup\{\, x^* \mid x \in X_1 \,\}.\end{aligned}$$

Daß die Mengen auf den rechten Seiten in (ii) gerichtet sind, folgt aus der leicht zu zeigenden Eigenschaft monotoner Funktionen:

Proposition 6.4 Ist $\mathcal{K}$ eine stetige Kleene'sche Algebra, $f : K^n \to K$ monoton und sind $X_1,\dots,X_n \subseteq K$ gerichtet, so ist auch $\{\, f(x_1,\dots,x_n) \mid x_1 \in X_1,\dots,x_n \in X_n \,\}$ gerichtet.

Proposition 6.5 In jeder stetigen Kleene'schen Algebra $\mathcal{K}$ ist $a^* = \bigsqcup\{\, \Sigma_{i=0}^n a^i \mid n \in I\!N \,\}$ für jedes $a \in K$, wobei $a^0 := 1$ und $a^{n+1} := a^n \cdot a$.

Beweis: Sei $a \in K$ und $X_a := \{\, \Sigma_{i=0}^n a^i \mid n \in I\!N \,\}$. Offenbar ist X_a eine gerichtete Menge. Da $\cdot$ und $+$ stetig sind, ist also

$$(a \cdot \bigsqcup X_a) + 1 \;=\; \bigsqcup(a \cdot X_a + 1) = \bigsqcup\{\, \Sigma_{i=0}^n a \cdot a^i + 1 \mid n \in I\!N \,\} = \bigsqcup X_a,$$

woraus $a^* \le \bigsqcup X_a$ mit *(KA-1)* folgt. Die Umkehrung $\bigsqcup X_a \le a^*$ ist klar, da man induktiv $a^n \le a^*$ aus *(KA-1)* erhält, also $\Sigma_{i=0}^n a^i \le a^*$ mit der Idempotenz der Addition. □

Jede obere Schranke von $\{\, a^n \mid n \in I\!N \,\}$ ist eine obere Schranke von $\{\, \Sigma_{i=0}^n a^i \mid n \in I\!N \,\}$ und umgekehrt; daher hat man auch $\bigsqcup\{\, a^n \mid n \in I\!N \,\} = a^*$.

Beispiel 6.6 Die Sprachinterpretation $\mathcal{L} = (2^{\Sigma^*}, \cup, \emptyset, \cdot, \{\varepsilon\}, {}^*, \{a\})_{a\in\Sigma}$ und die Relationeninterpretationen $\mathcal{R} = (2^{M\times M}, \cup, \emptyset, \circ, 1_M, {}^*, \vdash_a)_{a\in\Sigma}$ sind stetige Kleene-Algebren, wenn man $\bigsqcup X := \bigcup X$ für die kleinste obere Schranke einer Familie X von Sprachen bzw. Relationen nimmt.

Warnung: *Die Kleene'sche Algebra $Reg_\Sigma^{\mathcal{L}} \subset \mathcal{L}_\Sigma$ aller regulären Sprachen über Σ ist nicht stetig*: die Familie $X = \{\, \{\, a^k b^k \mid k \leq n \,\} \mid n \in I\!N \,\}$ von endlichen, also regulären Sprachen ist gerichtet, hat aber keine kleinste obere Schranke. Nach Korollar 4.21 ist nämlich $\bigcup X = \{\, a^n b^n \mid n \in I\!N \,\}$ keine reguläre Sprache. Jede reguläre obere Schranke R von X ist daher eine echte Obermenge von $\bigcup X$, und mit $w \in R - \bigcup X$ ist $R - \{w\}$ eine kleinere reguläre (!) obere Schranke von X.

Lemma 6.7 Sei $\mathcal{K}$ eine stetige Kleene'sche Algebra. Dann ist auch $\mathcal{M}_{n,n}(K)$, die Algebra der $n \times n$-Matrizen über $\mathcal{K}$, eine stetige Kleene'sche Algebra.

Beweis: Nach Satz 5.5 ist $\mathcal{M}_{n,n}(\mathcal{K})$ eine Kleene'sche Algebra. Da die Addition von Matrizen komponentenweise erfolgt, bedeutet $(a_{i,j}) \leq^{\mathcal{M}_{n,n}(\mathcal{K})} (b_{i,j})$, daß $a_{i,j} \leq^{\mathcal{K}} b_{i,j}$ für alle i, j gilt. Daraus folgt, daß die durch

$$(\bigsqcup X)_{i,j} = \bigsqcup\{\, A_{i,j} \mid A \in X \,\}$$

definierte Matrix $\bigsqcup X$ die kleinste obere Schranke der gerichteten Menge X von Matrizen ist, so daß $\leq^{\mathcal{M}_{n,n}(\mathcal{K})}$ eine vollständige partielle Ordnung auf $M_{n,n}(\mathcal{K})$ ist. Man sieht leicht, daß $+$ und $\cdot$ stetig und * monoton ist. Für die Iteration überprüfen wir

$$(\bigsqcup X)^* = \bigsqcup\{\, A^* \mid A \in X \,\}.$$

Sei $X^* := \{\, A^* \mid A \in X \,\}$ und X gerichtet. Mit der Stetigkeit von $+, \cdot$ und der Monotonie von * erhält man

$$\begin{array}{rcll}
(\bigsqcup X) \cdot (\bigsqcup X^*) + 1_n & = & \bigsqcup\{\, A \cdot B^* \mid A, B \in X \,\} + 1_n & \\
& = & \bigsqcup\{\, CC^* + 1 \mid C \in X \,\} & \text{da } X \text{ gerichtet ist} \\
& = & \bigsqcup\{\, C^* \mid C \in X \,\} & \\
& = & \bigsqcup X^*. &
\end{array}$$

Damit ist $\bigsqcup X^*$ eine Lösung von $Y = (\bigsqcup X)Y + 1$ und folglich $(\bigsqcup X)^* \leq \bigsqcup X^*$. Andererseits ist für $A \in X$ wegen $A \leq \bigsqcup X$ und der Monotonie von * auch $A^* \leq (\bigsqcup X)^*$, und daher $\bigsqcup X^* \leq (\bigsqcup X)^*$. □

Bemerkung 6.8 Wenn $\mathcal{K}$ stetig ist, kann man die Iterierte einer Matrix $A \in \mathcal{M}_{n,n}(K)$ auch wie folgt definieren. Wir betrachten das Gleichungssystem

$$x_{i,j} = \Sigma_{k=1}^n \; a_{i,k} \cdot x_{k,j} + \delta_{i,j} \qquad (1 \leq i, j \leq n) \tag{34}$$

mit $\delta_{i,j} := 0$ für $i \neq j$ und $\delta_{i,j} := 1$ für $i = j$, das der Matrixgleichung $X = A \cdot X + 1_n$ entspricht. Nach dem folgenden Satz existiert in $\mathcal{K}$ eine kleinste Lösung von (34) durch Elemente $b_{i,j} \in K$, $1 \leq i, j \leq n$. Nun sei $A^* := (b_{i,j}) \in M_{n,n}(\mathcal{K})$.

Seien $a_1, \ldots, a_n \in K$ Objekte zur Interpretation der Buchstaben $a_i \in \Sigma$, und sei

$$S = \begin{cases} x_1 & = & r_1(x_1, \ldots, x_m) \\ \vdots & \vdots & \vdots \\ x_m & = & r_m(x_1, \ldots, x_m) \end{cases}$$

ein reguläres Gleichungssystem über Σ. Dem System S ordnen wir durch die Festlegung

$$F_S(c_1, \ldots, c_m) := (r_1^{\mathcal{K}}(c_1, \ldots, c_m), \ldots, r_m^{\mathcal{K}}(c_1, \ldots, c_m))$$

eine Funktion $F_S : K^m \to K^m$ zu. Eine Lösung von S in $\mathcal{K}$ oder ein Fixpunkt von F_S ist ein Tupel $(c_1, \ldots, c_m)$ mit $F_S(c_1, \ldots, c_m) = (c_1, \ldots, c_m)$.

Satz 6.9 (Fixpunktsatz) Sei $\mathcal{K}$ eine stetige Kleene'sche Algebra mit Elementen $a_1 \in K$, $\ldots, a_n \in K$ zur Interpretation der Konstanten $a_1, \ldots, a_n \in \Sigma$. Jedes reguläre System

$$S = \begin{cases} x_1 & = & r_1(x_1, \ldots, x_m) \\ \vdots & \vdots & \vdots \\ x_m & = & r_m(x_1, \ldots, x_m) \end{cases}$$

über Σ hat eine bezüglich $\leq$ kleinste Lösung $(c_1, \ldots, c_m)$ in $\mathcal{K}$.

Beweis: Wir beweisen die Behauptung etwas allgemeiner, nämlich für stetige Funktionen $r_i : K^m \to K$. Beachte, daß in einer stetigen Kleene'schen Algebra die Operationen $+^{\mathcal{K}}$, $\cdot^{\mathcal{K}}$ und $^{*\mathcal{K}}$ nach Voraussetzung und daher jede durch einen regulären Ausdruck $r(x_1, \ldots, x_m)$ definierte Funktion $(b_1, \ldots, b_m) \mapsto r^{\mathcal{K}}(b_1, \ldots, b_m)$ nach Induktion über r stetig ist.

Wir definieren induktiv eine Folge von Tupeln $(c_{1,i}, \ldots, c_{m,i}) \in K^m$ durch

$$(c_{1,0}, \ldots, c_{m,0}) := (0, \ldots, 0) \qquad \text{und} \qquad (c_{1,i+1}, \ldots, c_{m,i+1}) := F_S(c_{1,i}, \ldots, c_{m,i}).$$

Da alle r_j stetig sind, ist F_S monoton auf $\mathcal{K}^m$. Wegen $(c_{1,0}, \ldots, c_{m,0}) \leq (c_{1,1}, \ldots, c_{m,1})$ ist daher $(c_{1,i}, \ldots, c_{m,i}) \leq (c_{1,i+1}, \ldots, c_{m,i+1})$ für alle $i \in I\!N$.

Beh.: $(c_1, \ldots, c_m) := (\bigsqcup\{ c_{1,i} \mid i \in I\!N \}, \ldots, \bigsqcup\{ c_{m,i} \mid i \in I\!N \})$ ist der kleinste Fixpunkt von F_S.

Aus der Stetigkeit der r_j erhält man

$$\begin{aligned} r_j(c_1, \ldots, c_m) &= r_j(\bigsqcup\{ c_{1,i} \mid i \in I\!N \}, \ldots, \bigsqcup\{ c_{m,i} \mid i \in I\!N \}) \\ &= \bigsqcup\{ r_j(c_{1,i_1}, \ldots, c_{m,i_m}) \mid i_1, \ldots, i_m \in I\!N \} \\ &= \bigsqcup\{ r_j(c_{1,i}, \ldots, c_{m,i}) \mid i \in I\!N \} \\ &= \bigsqcup\{ c_{j,i+1} \mid i \in I\!N \} = c_j, \end{aligned}$$

und folglich $F_S(c_1, \ldots, c_m) = (r_1(c_1, \ldots, c_m), \ldots, r_m(c_1, \ldots, c_m)) = (c_1, \ldots, c_m)$.

Es bleibt zu zeigen, daß dieser Fixpunkt unterhalb jedes anderen Fixpunkts liegt. Sei also $(d_1,\ldots,d_m) = F_S(d_1,\ldots,d_m)$. Dann ist $c_{j,0} = 0 \leq d_j$ für jedes j, und mit $(c_{1,k},\ldots,c_{m,k}) \leq (d_1,\ldots,d_m)$ hat man wegen der Monotonie von F_S auch

$$(c_{1,k+1},\ldots,c_{m,k+1}) = F_S(c_{1,k},\ldots,c_{m,k}) \leq F_S(d_1,\ldots,d_m) = (d_1,\ldots,d_m).$$

Also sind die d_j obere Schranken von $\{\, c_{j,k} \mid k \in I\!N \,\}$ und liegen daher oberhalb der kleinsten oberen Schranken, den c_j. Damit ist $(c_1,\ldots,c_m) \leq (d_1,\ldots,d_m)$ gezeigt. □

Die Argumentation gilt auch, wenn die Ausdrücke r_j weitere Variable $x_{m+1},\ldots,x_{m+k}$ enthalten, die wir durch fest gewählte Elemente $b_1,\ldots,b_k \in K$ interpretieren. Die kleinste Lösung eines solchen Systems

$$\begin{array}{rcl} x_1 & = & r_1(x_1,\ldots,x_m,x_{m+1},\ldots,x_{m+k}) \\ \vdots & \vdots & \vdots \\ x_m & = & r_m(x_1,\ldots,x_m,x_{m+1},\ldots,x_{m+k}) \end{array}$$

hängt dann natürlich von den Parametern $b_1,\ldots,b_k \in K$ ab. Zur Bezeichnung dieser kleinsten Lösung führen wir neue Ausdrücke der Form

$$\mu(x_1,\ldots,x_m).(r_1,\ldots,r_m)$$

ein. Diese enthalten dann Variable $x_{m+1},\ldots,x_{m+k}$ "frei", aber $x_1,\ldots,x_m$ werden durch den Fixpunkt-Operator μ "gebunden" und stehen z.B. nicht mehr für Einsetzungen zur Verfügung. Für feste $b_1,\ldots,b_k \in K$ wäre also

$$(\mu(x_1,\ldots,x_m).(r_1,\ldots,r_m))^{\mathcal{K}}(b_1,\ldots,b_k)$$

das $\leq$-kleinste Tupel $(c_1,\ldots,c_m)$ mit

$$\begin{array}{rcl} c_1 & = & r_1^{\mathcal{K}}(c_1,\ldots,c_m,b_1,\ldots,b_k) \\ \vdots & \vdots & \vdots \\ c_m & = & r_m^{\mathcal{K}}(c_1,\ldots,c_m,b_1,\ldots,b_k). \end{array}$$

Damit die neuen Ausdrücke —wie die bisherigen— nur Elemente von $\mathcal{K}$ und nicht Tupel von Elementen bezeichnen, könnte man Ausdrücke

$$\mu_i(x_1,\ldots,x_m).(r_1,\ldots,r_m)$$

zur Bezeichnung der i-ten Komponente der kleinsten Lösung verwenden. Stattdessen wählen wir nur einfache Gleichungen $x = r$ und einen einstelligen Fixpunktoperator, lassen aber zu, daß dieser in r auch schon vorkommen darf:

Definition 6.10 Die regulären Ausdrücke über Σ mit Fixpunktoperator, oder kurz die μ-regulären Ausdrücke über Σ, seien induktiv wie folgt definiert:

(i) Variable $x_1, x_2, \ldots$ und Konstante $a \in \Sigma$ sind μ-reguläre Ausdrücke über Σ.

(ii) Sind r_1 und r_2 μ-reguläre Ausdrücke über Σ, so auch $(r_1 + r_2)$, $(r_1 \cdot r_2)$ und r_1^*.

(iii) Ist x eine Variable und r eine μ-regulärer Ausdruck über Σ, so ist auch $\mu x.r$ ein μ-regulärer Ausdruck über Σ.

Der nächste Satz wird zeigen, daß diese Ausdrücke ausreichen, um die Komponenten der kleinsten Lösungen von regulären Gleichungssystemen zu bezeichnen.

Proposition 6.11 Ist $g : \mathcal{K}^n \to \mathcal{K}$ stetig, so ist für jedes $i \in \{1, \ldots, n\}$ auch die durch

$$h(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) := \mu y.g(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_n)$$

definierte Funktion $h : \mathcal{K}^{n-1} \to \mathcal{K}$ stetig.

Beweis: Sei etwa $i = 1$ und $n = 2$. Definiere einstellige Funktionen g_n und g_∞ durch

$$g_0(x) := g(x, 0), \qquad g_{n+1}(x) := g(x, g_n(x)), \qquad \text{und} \qquad g_\infty(x) := \bigsqcup\{\, g_n(x) \mid n \in I\!N \,\}.$$

Wegen der Stetigkeit ist g eine monotone Funktion in jedem seiner Argumente, und daher bekommen wir mit Induktion $g_n(x) \leq g_{n+1}(x)$, für alle n. Nach dem Fixpunktsatz ergibt das $\mu y.g(x, y) = g_\infty(x)$. Die Stetigkeit von g_∞ folgt aber aus der der g_n über

$$\begin{aligned} g_\infty(\bigsqcup X) &= \bigsqcup\{\, g_n(\bigsqcup X) \mid n \in I\!N \,\} \\ &= \bigsqcup\{\, \bigsqcup\{\, g_n(x) \mid x \in X \,\} \mid n \in I\!N \,\} \\ &= \bigsqcup\{\, g_n(x) \mid x \in X, n \in I\!N \,\} \\ &= \bigsqcup\{\, \bigsqcup\{\, g_n(x) \mid n \in I\!N \,\} \mid x \in X \,\} \\ &= \bigsqcup\{\, g_\infty(x) \mid x \in X \,\}. \end{aligned}$$

Damit ist auch $\mu y.g(x, y)$ stetig. □

Lemma 6.12 (Bekič/Scott-Prinzip) Seien $f(x, y, z_1, \ldots, z_k)$ und $g(x, y, z_1, \ldots, z_k)$ stetige Funktionen auf einer stetigen Kleene'schen Algebra $\mathcal{K}$ und $c_1, \ldots, c_k$ eine Folge von Elementen aus K. Für $a \in K$ sind äquivalent:

(i) Es gibt $b \in K$ so daß (a, b) die kleinste Lösung des Systems

$$\begin{aligned} x &= f(x, y, z_1, \ldots, z_k) \\ y &= g(x, y, z_1, \ldots, z_k) \end{aligned}$$

in $\mathcal{K}$ ist, wenn man die $z_1, \ldots, z_k$ durch $c_1, \ldots, c_k$ interpretiert.

(ii) Das Element a ist die kleinste Lösung der Gleichung

$$x = f(x, \mu y.g(x, y, z_1, \dots, z_k), z_1, \dots, z_k)$$

in $\mathcal{K}$, wenn man die $z_1, \dots, z_k$ durch $c_1, \dots, c_k$ interpretiert.

Beweis: Wir unterdrücken die $z_1, \dots, z_k$ und $c_1, \dots, c_k$ und schreiben z.B. einfach $f(a, b)$ statt $f(a, b, c_1, \dots, c_k)$. Es genügt zu zeigen, daß

$$\mu_1(x, y).(f(x, y), g(x, y)) \;=\; \mu x.f(x, \mu y.g(x, y)).$$

Beh. 1: $\mu_1(x, y).(f(x, y), g(x, y)) \leq \mu x.f(x, \mu y.g(x, y))$.

Beweis: Wähle $(x_0, y_0) \in K \times K$ durch $x_0 := \mu x.f(x, \mu y.g(x, y))$ und $y_0 := \mu y.g(x_0, y)$ nach dem Fixpunktsatz. Nach Definition von μ haben wir

$$x_0 = f(x_0, \mu y.g(x_0, y)) = f(x_0, y_0) \quad \text{und} \quad y_0 = g(x_0, y_0),$$

weshalb das Paar (x_0, y_0) eine Lösung des Systems $\{x = f(x, y),\ y = g(x, y)\}$ ist. Vergleichen wir das mit der kleinsten Lösung, so erhalten wir

$$\mu(x, y).(f(x, y), g(x, y)) \leq (x_0, y_0).$$

Also ist $\mu_1(x, y).(f(x, y), g(x, y)) \leq x_0$, womit Beh. 1 gezeigt wäre.

Beh. 2: $\mu x.f(x, \mu y.g(x, y)) \leq \mu_1(x, y).(f(x, y), g(x, y))$.

Sei $(\bar{x}, \bar{y})$ die kleinste Lösung des Systems $\{x = f(x, y),\ y = g(x, y)\}$. Wenn wir

$$\bar{y} = \mu y.g(\bar{x}, y), \tag{$*$}$$

zeigen können, haben wir $\bar{x} = f(\bar{x}, \bar{y}) = f(\bar{x}, \mu y.g(\bar{x}, y))$. Durch einen Vergleich mit der kleinsten Lösung von $x = f(x, \mu y.g(x, y))$ sieht man nun, daß Beh. 2 gilt:

$$\mu x.f(x, \mu y.g(x, y)) \leq \bar{x} = \mu_1(x, y).(f(x, y), g(x, y)).$$

Um die Gleichung $(*)$ zu zeigen, beachte man, daß $\mu y.g(\bar{x}, y) \leq \bar{y}$ direkt aus $\bar{y} = g(\bar{x}, \bar{y})$ folgt. Für die andere Ungleichung,

$$\bar{y} \leq \mu y.g(\bar{x}, y),$$

definieren wir eine Folge $(x_n, y_n) \in K \times K$ durch

$$(x_0, y_0) := (0, 0), \qquad \text{und} \qquad (x_{n+1}, y_{n+1}) := (f(x_n, y_n), g(x_n, y_n)).$$

Wegen der Monotonie von f und g folgt aus $(0,0) \leq (x_1, y_1)$ schon $(x_n, y_n) \leq (x_{n+1}, y_{n+1})$ für alle n. Nach dem Fixpunktsatz haben wir $(\bar{x}, \bar{y}) = (\bigsqcup_n x_n, \bigsqcup_n y_n)$. Die Monotonie von g und $x_n \leq \bar{x}$ ergeben auch

$$y_{n+1} = g(x_n, y_n) \leq g(\bar{x}, y_n).$$

Daher ist $y_1 \leq g(\bar{x}, y_0) = g(\bar{x}, 0)$. Für die durch

$$g_0(x) := g(x, 0), \qquad g_{n+1}(x) := g(x, g_n(x)), \qquad \textit{und} \qquad g_\infty(x) := \bigsqcup \{\, g_n(x) \mid n \in I\!N \,\}$$

definierten Funktionen g_n und g_∞ ist dann $y_1 \leq g_0(\bar{x})$, und induktiv erhält man aus $y_n \leq g_{n-1}(\bar{x})$, daß

$$y_{n+1} \leq g(\bar{x}, y_n) \leq g(\bar{x}, g_{n-1}(\bar{x})) = g_n(\bar{x}).$$

Das ergibt $\bar{y} = \bigsqcup_n y_n \leq \bigsqcup_n g_n(\bar{x}) = g_\infty(\bar{x}) = \mu y.g(\bar{x}, y)$, und Gleichung $(*)$ ist gezeigt. $\square$

Aus Proposition 6.11 folgt, daß wir jeden μ-regulären Ausdruck – auch wenn μ geschachtelt vorkommt – in einer stetigen Kleene'schen Algebra $\mathcal{K}$ interpretieren können, indem wir

$$(\mu x.r(x, y_1, \ldots, y_n))^{\mathcal{K}}(b_1, \ldots, b_n) \quad := \quad \text{das kleinste } b \in K \text{ mit } b = r^{\mathcal{K}}(b, b_1, \ldots, b_n)$$

setzen. Für kleinste Lösungen von regulären Gleichungssystemen gilt dann:

Satz 6.13 Sei $\mathcal{K}$ eine stetige Kleene'sche Algebra und $a \in K$. Dann sind äquivalent:

(i) Das Element a ist die erste Komponente der kleinsten Lösung in $\mathcal{K}$ eines regulären Gleichungssystems

$$\begin{array}{ccc} x_1 & = & r_1(x_1, \ldots, x_m) \\ \vdots & \vdots & \vdots \\ x_m & = & r_m(x_1, \ldots, x_m). \end{array}$$

(ii) Es gibt einen μ-regulären Ausdruck s ohne freie Variable, so daß $a = s^{\mathcal{K}}$.

Beweis: Durch Induktion über die Anzahl m der Gleichungen zeigen wir die Behauptung für alle Gleichungssysteme mit μ-regulären (!) $r_j(x_1, \ldots, x_m)$ in (i). Kleinste Lösungen solcher Systeme gibt es nach Proposition 6.11 und dem Beweis des Fixpunktsatzes.

$m = 1$: Die Behauptung gilt mit $s := \mu\, x_1.r_1(x_1)$.

$m + 1$: Wir lassen die letzte Gleichung $x_{m+1} = r_{m+1}(x_1, \ldots, x_{m+1})$ weg und ersetzen in den übrigen die Variable x_{m+1} durch den Ausdruck $(\mu\, x_{m+1}.r_{m+1})(x_1, \ldots, x_m)$. Dadurch erhalten wir ein Gleichungssystem

$$\begin{array}{ccc} x_1 & = & r'_1(x_1, \ldots, x_m) \\ \vdots & \vdots & \vdots \\ x_m & = & r'_m(x_1, \ldots, x_m) \end{array}$$

mit nur m Gleichungen. Nach Induktionsannahme ist die erste Komponente seiner kleinsten Lösung schon der Wert eines μ-regulären Ausdrucks s ohne freie Variable.

Nach dem Lemma 6.12 von Bekič/Scott (für $m+1$ statt nur zwei Gleichungen) stimmt aber die kleinste Lösung des neuen Gleichungssystems mit den ersten m Komponenten der kleinsten Lösung des Ausgangssystems überein. □

6.2 Reguläre Gleichungssysteme als Sprachen, Relationen und Programme

Nach dem Fixpunktsatz 6.9 hat jedes reguläre Gleichungssystem

$$\begin{array}{rcl} x_1 & = & r_1(x_1,\ldots,x_m) \\ \vdots & \vdots & \vdots \\ x_m & = & r_m(x_1,\ldots,x_m) \end{array}$$

über Σ eine kleinste Lösung in jeder stetigen Kleene'schen Algebra, und daher nach Beispiel 6.6 insbesondere in der Sprachinterpretation

$$\mathcal{L} = (2^{\Sigma^*}, \cup, \emptyset, \cdot, \{\varepsilon\}, {}^*, \{a\})_{a\in\Sigma}$$

und in den Relationenalgebren

$$\mathcal{R} = (2^{M\times M}, \cup, \emptyset, \circ, 1_M, {}^*, \vdash_a)_{a\in\Sigma}$$

mit vorgegebenen Relationen $\vdash_a \subseteq M \times M$. Wir sagen, eine Sprache $L \subseteq \Sigma^*$ bzw. eine Relation $R \subseteq M \times M$ sei GLEICHUNGSDEFINIERBAR, wenn L bzw. R eine Komponente der kleinsten Lösung in $\mathcal{L}$ bzw. $\mathcal{R}$ eines regulären Gleichungssystems über Σ ist.

Im Unterschied z.B. zu den Operationen $\cap$, $-$, $\setminus$ und $/$ können wir den kleinsten Fixpunktoperator μ wie die regulären Operationen $+$, $\cdot$ und * als ein Programmkonstrukt verstehen, die REKURSION:

$$\mu x.r \quad \simeq \quad \textit{führe den Befehl } x = r(x) \textit{ rekursiv aus.}$$

Das Programm $\mu x.r$ arbeitet wie r, wobei es an Stelle des Befehls x sich selbst aufruft. Entsprechend kann man ein reguläres Gleichungssystem als eine Ansammlung mehrerer wechselseitig rekursiver Programme auffassen.

Man kann nun den Zusammenhang zwischen operationaler und denotationeller Bedeutung von Programmen aus Satz 2.5 auf rekursive Programme ausdehnen. Wir hatten jeder Sprache $L \subseteq \Sigma^*$ durch

$$\vdash_L := \bigcup\{\vdash_w \mid w \in L\}, \qquad \text{mit } \vdash_\varepsilon := 1_M, \qquad \vdash_{va} := \vdash_v \circ \vdash_a,$$

eine Relation $\vdash_L \subseteq M \times M$ zugeordnet.

Satz 6.14 Sei $r(x_1, \ldots, x_m)$ ein μ-regulärer Ausdruck über Σ, $L_1, \ldots, L_m \subseteq \Sigma^*$ eine Folge von Sprachen und $\vdash_{L_1}, \ldots, \vdash_{L_m} \subseteq M \times M$ die entsprechenden Relationen von $\mathcal{R}$. Dann gilt

$$\vdash_{r^{\mathcal{L}}(L_1,\ldots,L_m)} \;=\; r^{\mathcal{R}}(\vdash_{L_1}, \ldots, \vdash_{L_m}).$$

Auf den Beweis des Satzes wollen wir hier verzichten. Er besagt, daß die durch ein rekursives Programm r erzeugte Relation zwischen Ausgangs- und Endzuständen des Speichers einer Maschine gerade die auf dieselbe rekursive Weise r aus den Grundrelationen $\vdash_a$ definierte Relation ist. (Die L_i und $\vdash_{L_i}$ kann man als operationale und denotationelle Bedeutung von "Unterprogrammen" oder "Orakel" x_i verstehen.) Da $\mu x.r$ eine Lösung von $x = r(x)$ bezeichnet, gilt auch

$$\vdash_{(\mu x.r)^{\mathcal{L}}} \quad = \quad \vdash_{(r[\mu x.r/x])^{\mathcal{L}}},$$

und das ist die aus der Programmierung bekannte "Kopierregel": bei einem rekursiven Aufruf x eines Programms wird eine Kopie des Programms $\mu x.r$ ausgeführt.

Der Satz gilt aber nicht für die anderen genannten Operationen; man hat nur:

Proposition 6.15 Für beliebige $L_i \subseteq \Sigma^*$ und zugeordnete $\vdash_{L_i}$ in $\mathcal{R}$ ist

$$\begin{array}{lclclcl} \vdash_{L_1 - L_2} & \supseteq & \vdash_{L_1} - \vdash_{L_2} & \qquad & \vdash_{L_1 \cap L_2} & \subseteq & \vdash_{L_1} \cap \vdash_{L_2} \\ \vdash_{L_1 / L_2} & \subseteq & \vdash_{L_1} / \vdash_{L_2} & \qquad & \vdash_{L_1 \setminus L_2} & \subseteq & \vdash_{L_1} \setminus \vdash_{L_2}. \end{array}$$

Daß hier im Allgemeinen keine Gleichheiten gelten, kann man als Bestätigung dafür auffassen, daß diese Operationen keine Deutung als Programmkonstrukte haben.

Aus Satz 6.14 folgt wieder, daß die Sprachinterpretation feiner ist als die Relationeninterpretation:

Korollar 6.16 Für μ-reguläre Ausdrücke r und s ohne freie Variable gilt

$$\mathcal{L} \models r = s \quad \Rightarrow \quad \mathcal{R} \models r = s.$$

Eine eingeschränkte Form der Rekursion kann schon durch die Iteration ausgedrückt werden, wie wir in Lemma 3.7 gezeigt haben:

Proposition 6.17 In jeder Kleene'schen Algebra $\mathcal{K}$ gilt für alle $a, b \in K$

$$\mu x(ax + b) = a^* b \qquad \text{und} \qquad \mu x(xa + b) = ba^*.$$

Die Gleichung $\mu x(ax + b) = a^*b$ bedeutet in der Programmierung, daß ENDREKURSIVE Programme, d.h. solche, bei denen Rekursionsaufrufe nur als letzte Programmschritte vorkommen, in ITERATIVE Programme umgewandelt werden können. Das ist von praktischer Bedeutung, da man zur Implementierung der Iteration keinen Kellerspeicher braucht — im Unterschied zur Implementierung der allgemeinen Rekursion. (vgl. Beispiel 7.4)

Für die Definition von Sprachen bedeutet $\mu x(ax + b) = a^*b$, daß rechtslineare Grammatiken (vgl. Abschnitt 4.2) nur reguläre Sprachen definieren.

Die Gleichungen aus 6.17 gelten in angepaßter Form auch dann, wenn a und b (in μ-regulärer Weise) von x abhängen. Diese Verallgemeinerung ist nützlich, wenn man Linksrekursionen in Sprachdefinitionen beseitigen will (vgl. Abschnitt 6.4).

Lemma 6.18 In jeder stetigen Kleene'schen Algebra $\mathcal{K}$ gilt

$$\forall \vec{z} \ (\ \mu x(rx + s) = \mu x(r^*s) \ \wedge \ \mu x(xr + s) = \mu x(sr^*) \ )$$

für alle μ-regulären Ausdrücke $r(x, \vec{z})$ und $s(x, \vec{z})$.

Beweis: Sei y eine noch nicht verwendete Variable. Die Lösungen von $x = rx + s$ sind gerade die x-Komponenten der Lösungen des Systems

$$(*) \qquad \begin{array}{rcl} x & = & y \qquad\qquad =: \ f(x, y, \vec{z}) \\ y & = & ry + s \quad\ =: \ g(x, y, \vec{z}). \end{array}$$

in $\mathcal{K}$ (relativ zu Elementen $\vec{c}$ zur Interpretation der freien Variablen $\vec{z}$). Insbesondere ist also für die kleinste Lösung $(\bar{x}, \bar{y})$ von $(*)$ nach Lemma 6.12

$$\begin{array}{rcl} \mu x\,(xr + s) & = & \bar{x} \\ & = & \mu x.f(x, \mu y.g(x, y, \vec{z}), \vec{z}) \\ & = & \mu x\,\mu y.g(x, y, \vec{z}) \\ & = & \mu x\,\mu y\,(ry + s). \end{array}$$

Nach Proposition 6.17 erfüllt $\mathcal{K}$ aber $\forall \vec{z} \forall x(\mu y(ry + s) = r^*s)$, womit die erste Gleichung $\mu x(rx + s) = \mu x(r^*s)$ folgt. Analog ist die zweite Gleichung zu zeigen. □

6.3 Umformung von Gleichungssystemen

Simultan induktive Definitionen von Mengen, die mit den regulären Operationen erzeugt werden, lassen sich kurz durch reguläre Gleichungssysteme ausdrücken. Die μ-regulären Ausdrücke traten in natürlicher Weise als Lösungsterme regulärer Gleichungssysteme auf und hätten nach dem Beweis von Satz 6.13 auch in der rechten Seiten der Gleichungssysteme zugelassen werden können.

Wir wollen nun umgekehrt die in den Gleichungssystemen auftretenden Terme vereinfachen, also reguläre Gleichungssysteme nach gewissen Regeln umformen, ohne daß sich die Hauptkomponente der kleinsten Lösung ändert.

Definition 6.19 Sei $\mathcal{K}$ eine stetige Kleene'sche Algebra. Zwei reguläre Gleichungssysteme

$$R = \begin{cases} x_1 & = & r_1(x_1,\ldots,x_m) \\ \vdots & \vdots & \vdots \\ x_m & = & r_m(x_1,\ldots,x_m) \end{cases} \quad \text{und} \quad S = \begin{cases} x_1 & = & s_1(x_1,\ldots,x_k) \\ \vdots & \vdots & \vdots \\ x_k & = & s_k(x_1,\ldots,x_k) \end{cases}$$

in den Rekursionsvariablen $x_1,\ldots,x_m$ heißen $\mathcal{K}$-ÄQUIVALENT (bezüglich einer Belegung $c_1,\ldots,c_l \in K$ weiterer Variablen $y_1,\ldots,y_l$), falls die kleinsten Lösungen $(a_1,\ldots,a_m)$ von R und $(b_1,\ldots,b_k)$ von S in der Hauptkomponente übereinstimmen, d.h. $a_1 = b_1$.

Eine Variable x_j KANN IN R VON x_i ERREICHT WERDEN, falls $i = j$ oder falls x_j in R von einer in r_i frei vorkommenden Variablen erreicht werden kann.

Lemma 6.20 Das reguläre Gleichungssystem S über $\mathcal{K}$ entstehe aus R durch eine der folgenden Transformationen. Dann sind R und S äquivalent.

(i) *Austausch von in KA äquivalenten Ausdrücken*: Ersetze ein oder mehrere Vorkommen von $r(x_1,\ldots,x_m)$ in R durch einen Ausdruck $s(x_1,\ldots,x_m)$ mit $KA \vdash r = s$.

(ii) *Beseitigung der Iteration*: Ersetze einen Teilausdruck r^* aus R durch eine neue Variable x_{m+1} und füge eine Gleichung $x_{m+1} = r \cdot x_{m+1} + 1$ zum System hinzu.

(iii) *Beseitigung von scheinbaren Rekursionsvariablen*: Kommt die Rekursionsvariable x_m in r_m nicht vor und ist $m > 1$, so entferne die Gleichung $x_m = r_m$ aus R und ersetze x_m in den übrigen Gleichungen überall durch r_m.

(iv) *Beseitigung von Variablenzyklen*: Bilden die Variablen $x_k,\ldots,x_m$ einen Zyklus, d.h. haben die letzten $m-k$ Gleichungen die Form

$$x_k = r'_k + x_{k+1}, \quad \ldots, \quad x_{m-1} = r'_{m-1} + x_m, \quad x_m = r'_m + x_k,$$

so ersetze diese durch $x_k = r'_k + \ldots + r'_m$ und $x_{k+1},\ldots,x_m$ durch x_k.

(v) *Beseitigung von unerreichbaren Gleichungen*: Ist x_m von x_1 nicht erreichbar, so entferne die Geichung $x_m = r_m$.

(vi) *Einführung scheinbarer Rekursionsvariablen*: Ersetze Vorkommen von $r(x_1,\ldots,x_m)$ in R durch eine neue Variable x_{m+1} und füge die Gleichung $x_{m+1} = r$ hinzu.

Durch Umordnen der Gleichungen $x_2 = r_2,\ldots,x_m = r_m$ kann man stets erreichen, daß man nur die m-te bzw. die letzten k Gleichungen verändern muß.

Beweis:

(i) Da in $\mathcal{K}$ die Aussage $\forall x_1 \ldots \forall x_m (r = s)$ gilt, ist $r^{\mathcal{K}}(a_1, \ldots, a_m) = r^{\mathcal{K}}(a_1, \ldots, a_m)$ für alle $a_1, \ldots, a_m \in K$, also auch $r_i^{\mathcal{K}}(a_1, \ldots, a_m) = s_i^{\mathcal{K}}(a_1, \ldots, a_m)$ für alle i. Daher sind die kleinsten Lösungen von R und S dieselben.

(ii) Sei $(a_1, \ldots, a_m)$ von R die kleinste Lösung und $(b_1, \ldots, b_{m+1})$ die kleinste Lösung von S. Dann ist $b_{m+1} = r^{\mathcal{K}}(b_1, \ldots, b_m) \cdot b_{m+1} + 1$ die kleinste Lösung von $x = r^{\mathcal{K}}(b_1, \ldots, b_m) \cdot x + 1$, also wegen $\mathcal{K} \models KA$

$$b_{m+1} = (r^{\mathcal{K}}(b_1, \ldots, b_m))^* = (r^*)^{\mathcal{K}}(b_1, \ldots, b_m).$$

Daher löst $(b_1, \ldots, b_{m+1})$ das System $R[x_{m+1}/r^*]$ und folglich ist $(b_1, \ldots, b_m)$ eine Lösung von R, also $a_1 \leq b_1$. Umgekehrt ist mit $a_{m+1} := (r^*)^{\mathcal{K}}(a_1, \ldots, a_m)$ das Tupel $(a_1, \ldots, a_{m+1})$ eine Lösung von S, also $b_1 \leq a_1$, da $(b_1, \ldots, b_{m+1})$ die kleinste Lösung von S ist.

(iii) Sei $(a_1, \ldots, a_m)$ die kleinste Lösung von R, $b_1, \ldots, b_{m-1}$ die kleinste Lösung von S. Wegen $a_m = r_m^{\mathcal{K}}(a_1, \ldots, a_{m-1})$ ist für $i < m$

$$\begin{aligned} a_i &= r_i^{\mathcal{K}}(a_1, \ldots, a_m) = r_i^{\mathcal{K}}(a_1, \ldots, a_{m-1}, r_m^{\mathcal{K}}(a_1, \ldots, a_{m-1})) \\ &= r_i[r_m/x_m]^{\mathcal{K}}(a_1, \ldots, a_{m-1}) = s_i^{\mathcal{K}}(a_1, \ldots, a_{m-1}). \end{aligned}$$

Also ist $(a_1, \ldots, a_{m-1})$ eine Lösung von S und daher $b_1 \leq a_1$. Umgekehrt ist wegen $b_i = s_i^{\mathcal{K}}(b_1, \ldots, b_{m-1})$ mit $b_m := r_m^{\mathcal{K}}(b_1, \ldots, b_{m-1})$ auch $b_i = r_i^{\mathcal{K}}(b_1, \ldots, b_m)$ für $i < m$ und daher $(b_1, \ldots, b_m)$ eine Lösung von R, also $a_1 \leq b_1$.

(iv) Betrachte zunächst den Fall, wo die Gleichungen durch $x_k = r'_k + \ldots + r'_m + x_k$ statt durch $x_k = r'_k + \ldots + r'_m$ ersetzt werden.

Sei $(a_1, \ldots, a_m)$ die kleinste Lösung von R, $(b_1, \ldots, b_k)$ die kleinste Lösung von S. Dann ist $a_k \geq a_{k+1} \geq \ldots \geq a_m \geq a_k$ und folglich $a_k = \ldots = a_m$. Für $i < k$ ist also $a_i = r_i^{\mathcal{K}}(a_1, \ldots, a_m) = r_i^{\mathcal{K}}(a_1, \ldots, a_k, \ldots, a_k) = s_i^{\mathcal{K}}(a_1, \ldots, a_k)$. Außerdem ist

$$a_k \geq (r'_k + \ldots + r'_m + x_k)^{\mathcal{K}}(a_1, \ldots, a_m) \geq a_k = s_k^{\mathcal{K}}(a_1, \ldots, a_k).$$

Daher ist $(a_1, \ldots, a_k)$ eine Lösung von S und somit $b_1 \leq a_1$. Umgekehrt ist wegen $s_i \equiv r_i[x_k/x_{k+1}, \ldots, x_k/x_m]$ für $i < k$ das m-Tupel $(b_1, \ldots, b_k, \ldots, b_k)$ offenbar eine Lösung der Gleichungen $x_i = r_i$ für $i < k$. Außerdem ist wegen

$$s_k \equiv (r'_k + \ldots + r'_m + x_k)[x_k/x_{k+1}, \ldots, x_k/x_m]$$

zunächst $b_k = (r'_k + \ldots + r'_m + x_k)^{\mathcal{K}}(b_1, \ldots, b_k, \ldots, b_k)$ und folglich auch $b_k = (r'_{k+i} + x_{k+i})^{\mathcal{K}}(b_1, \ldots, b_k, \ldots, b_k)$. Also ist $(b_1, \ldots, b_k)$ eine Lösung von R und daher $a_1 \leq b_1$.

Es bleibt noch zu zeigen, daß wir den Summanden x_k in der Gleichung $x_k = r' + x_k$ mit $r' = r'_k + \ldots + r'_m$ weglassen können. Nach Proposition 6.18 sind für feste $(b_1, \ldots, b_{k-1})$ die kleinsten Lösungen von $x_k = r'(b_1, \ldots, b_{k-1}, x_k) + x_k$ und $x_k = r'(b_1, \ldots, b_{k-1}, x_k)$ in $\mathcal{K}$ gleich, da $\mu x\,(1 \cdot x + r') = \mu x.(1^* \cdot r') = \mu x.r'$. Nach dem Lemma 6.12 ändert sich die Hauptkomponente der kleinsten Lösung von S bei dieser Umformung nicht.

(v) Sei $(a_1,\ldots,a_m)$ die kleinste Lösung von R und $(b_1,\ldots,b_{m-1})$ die kleinste Lösung von S. Da $(a_1,\ldots,a_{m-1})$ eine Lösung von S ist, ist $b_1 \leq a_1$. Sei b_m die kleinste Lösung von $x_m = r_m$, wenn man $x_1,\ldots,x_{m-1}$ durch $b_1,\ldots,b_{m-1}$ interpretiert. Dann ist $(b_1,\ldots,b_m)$ eine Lösung von R, also $a_1 \leq b_1$.

(vi) Das geht umgekehrt wie bei der Beseitigung scheinbarer Rekursionsvariablen.

Die Stetigkeit von $\mathcal{K}$ sichert die Existenz einer kleinsten Lösung von R und S. □

Definition 6.21 Ein reguläres Gleichungssystem über Σ heiße NORMIERT, wenn es keine unerreichbaren Gleichungen enthält und von der Form

$$\begin{array}{rcl} x_1 & = & r_{1,1}(x_1,\ldots,x_m)+\ldots+r_{1,k_1}(x_1,\ldots,x_m) \\ \vdots & \vdots & \vdots \\ x_m & = & r_{m,1}(x_1,\ldots,x_m)+\ldots+r_{m,k_m}(x_1,\ldots,x_m) \end{array} \tag{35}$$

mit Summanden $r_{i,j} \in \{0,1\} \cup (\Sigma \cup X)^+ - X$ ist, für $X = \{x_1,\ldots,x_m\}$.

Lemma 6.22 Jedes reguläre Gleichungssystem über der stetigen Kleene'schen Algebra $\mathcal{K}$ kann in ein äquivalentes normiertes System überführt werden.

Beweis: Entferne mit (v) zuerst alle von der Hauptvariablen x_1 nicht erreichbaren Gleichungen. Dann beseitige man mit (ii) alle Vorkommen von *; dabei neu eingeführte Rekursionsvariable sind von x_1 aus erreichbar. Mit den Halbringeigenschaften können die rechten Seiten nach (i) ausmultipliziert werden, bis sie Summen von Produkten mit mindestens einem Faktor werden. Durch Einführung neuer Rekursionsvariablen (vi) wird dann erreicht, daß jeder Faktor atomar ist, d.h. 0,1,eine Rekursionsvariable x_i oder ein Parameter c (wie $a \in \Sigma$) ist. Nach (i) kann man mit $1 \cdot x_i = x_i = x_i \cdot 1$ und $0 \cdot x_i = 0 = x_i \cdot 0$ sowie $r + 0 = r = 0 + r$ und der Assoziativität von $+$ die Summen weiter vereinfachen; falls dadurch unerreichbare Gleichungen entstehen, läßt man sie nach (v) weg. Gleiche Summanden kann man mit (i) wegen der Idempotenz der Addition zusammenziehen.

Summanden der Form x_i lassen sich wie folgt eliminieren: Mit (iv) beseitigen wir alle Variablenzyklen, wobei man $x_k = x_j$ wie $x_k = 0 + x_j$ behandelt. Nun gibt es unter den verbliebenen Gleichungen der Form $x_k = r'_k + x_j$ mindestens eine, bei der r_j in $x_j = r_j$ keine Summanden x_i hat. Daher kann man $x_k = r'_k + x_j$ durch $x_k = r'_k + r_j$ ersetzen, wobei sich offenbar die kleinste Lösung nicht ändert. Durch Induktion über die Anzahl der Summanden x_j folgt, daß sie alle beseitigt werden können. Falls dadurch unerreichbare Gleichungen entstehen, läßt man sie nach (v) weg.

Weiter kann man Gleichungen $x_i = 0$ für $i > 1$ mit (iii) beseitigen, und mit den Beziehungen zwischen $0, +$ und $\cdot$ erreichen, daß 0 nur noch dann auftritt, wenn das System zu $x_1 = 0$ wird. □

6.4 Kontextfreie Grammatiken und ihre Normalformen

Wir beschränken uns nun auf die stetige Kleene'sche Algebra $\mathcal{L}_\Sigma$ aller Sprachen über Σ und betrachten einige weitere Umformungen regulärer Gleichungssysteme, die hier zusätzlich zu denen des vorigen Abschnitts möglich sind.

Jedes reguläre Gleichungssystem $P = \{ x_i = r_i \mid 1 \leq i \leq m \}$ mit Parametern aus Σ definiert in $\mathcal{L}_\Sigma$ ein Tupel $(A_1, \ldots, A_m)$ von Sprachen, den kleinsten Fixpunkt der durch P gegebenen Stadienfunktion

$$F_P(B_1, \ldots, B_m) := (r_1^{\mathcal{L}_\Sigma}(B_1, \ldots, B_m), \ldots, r_m^{\mathcal{L}_\Sigma}(B_1, \ldots, B_m)).$$

Solche simultan induktiv definierten Sprachen gibt man oft durch einen Spezialfall regulärer Gleichungssysteme an, die sogenannten kontextfreien Grammatiken:

Definition 6.23 Eine KONTEXTFREIE GRAMMATIK $G = (N, \Sigma, P, S)$ besteht aus

(i) einer endlichen Menge N von Variablen oder NONTERMINALSYMBOLEN,

(ii) einer endlichen Menge Σ von Konstanten oder TERMINALSYMBOLEN, mit $N \cup \Sigma = \emptyset$,

(iii) einer endlichen Menge $P \subseteq N \times (N \cup \Sigma)^*$ von REGELN oder PRODUKTIONEN, und

(iv) einer Hauptvariablen, SATZKATEGORIE oder dem Startsymbol $S \in N$.

Es ist üblich, Variable durch Großbuchstaben $A, B, \ldots$ zu bezeichnen und eine Regel $(A, \alpha) \in P$ in der Form $A \to \alpha$ zu schreiben. Wir benutzen $V := N \cup \Sigma$.

Für $\varrho, \psi \in V^*$ sagen wir, ψ IST AUS ϱ MIT G IN EINEM SCHRITT ERZEUGBAR, $\varrho \Longrightarrow_G \psi$, falls es eine Regel $A \to \beta \in P$ und $\varrho_1, \varrho_2 \in V^*$ gibt, so daß

$$\varrho \equiv \varrho_1 A \varrho_2 \quad \text{und} \quad \psi \equiv \varrho_1 \beta \varrho_2.$$

DIE VON G ERZEUGTE SPRACHE ist

$$L(G) := \{w \in \Sigma^* \mid S \overset{*}{\Longrightarrow}_G w\},$$

wobei $\overset{*}{\Longrightarrow}_G$ die reflexive, transitive Hülle von $\Longrightarrow_G$ ist. Zwei Grammatiken G_1 und G_2 heißen (schwach) ÄQUIVALENT, wenn $L(G_1) = L(G_2)$. Eine G-Ableitung von β aus α ist ein Weg $\alpha_0 \Longrightarrow_G \alpha_1 \Longrightarrow_G \ldots \Longrightarrow_G \alpha_m$ mit $\alpha \equiv \alpha_0$ und $\beta \equiv \alpha_m$.

Jedes reguläre Gleichungssystem

$$P = \begin{cases} x_1 & = & r_1(x_1, \ldots, x_m) \\ \vdots & \vdots & \vdots \\ x_m & = & r_m(x_1, \ldots, x_m), \end{cases}$$

ist eine kontextfreie Grammatik G, wenn seine rechten Seiten *-frei und in DISJUNKTIVER NORMALFORM sind, d.h wenn jedes r_i eine Summe $r_{i,1} + \ldots + r_{i,k_i}$ von Produkten $r_{i,j}$ der Faktoren $1, x_1, \ldots, x_m$ und der Parameter $a_1, \ldots, a_n$ ist: die Rekursionsvariablen sind die Nonterminalsymbole N, wobei x_1 der Satzkategorie S entspricht, die Parameter sind die Terminalsymbole Σ, und die Summanden $r_{i,j}$ von r_i werden als Regeln $x_i \to r_{i,j}$ aufgefaßt. Umgekehrt ist jede kontextfreie Grammatik ein reguläres Gleichungssystem in *-freier disjunktiver Normalform, außer daß N eine *Ober*menge der Rekursionsvariablen und Σ eine *Ober*menge der Parameter des Gleichungssystems sein kann.

Beispiel 6.24 Sei S die Hauptvariable im System

$$P = \left\{ \begin{array}{lcllcl} S & = & aC + A, & A & = & AB, \\ B & = & AbS + ab, & C & = & c + SC. \end{array} \right.$$

Eine entsprechende kontextfreie Grammatik $G = (N', \Sigma', P', S')$ hat die Regelmenge

$$P' = \left\{ \begin{array}{lcllcl} S & \to & aC, & A & \to & AB, \\ S & \to & A, & & & \\ B & \to & AbS, & C & \to & c, \\ B & \to & ab, & C & \to & SC \end{array} \right\}$$

und $S' \equiv S$, $N' \supseteq \{S, A, B, C\}$ und $\Sigma' \supseteq \{a, b, c\}$. Da

$$S \Longrightarrow_G aC \Longrightarrow_G aSC \Longrightarrow_G aSc \Longrightarrow_G aaCc \Longrightarrow_G aacc$$

eine mit der Satzkategorie beginnende G-Ableitung ist, ist $aacc \in L(G)$.

Wir identifizieren ein *-freies reguläres Gleichungssystem P in distributiver Normalform oft mit der Regelmenge P einer kontextfreien Grammatik und umgekehrt (wobei die in P nicht vorkommenden Variablen und Konstanten weggelassen werden, vgl. 6.29). Daß $L(G)$ tatsächlich die S-Komponente der kleinsten Lösung des Gleichungssystems P in $\mathcal{L}_\Sigma$ ist, folgt aus Satz 7.8 unten.

Eine ERWEITERT KONTEXTFREIE GRAMMATIK ist Grammatik $G = (N, \Sigma, P, S)$, bei der die rechten Seiten der Regeln von P reguläre Ausdrücke über $\Sigma \cup N$ sind. Wir identifizieren G wieder mit dem durch P gegebenen regulären Gleichungssystem.

Die Summanden oder rechten Regelseiten einer syntaktischen Kategorie x von G sind alternative AUSDRUCKSFORMEN oder Konstruktionsmöglichkeiten für Ausdrücke w der Kategorie x. Das Format der kontextfreien Grammatik ist nützlich, wenn man die Regeln numeriert, da dann auf jede Alternative eindeutig Bezug genommen werden kann. Eine syntaktische Analyse eines Wortes $w \in \Sigma^*$ kann dann durch eine Zahlenfolge dargestellt werden (vgl. Abschnitt 9.4).

Die Umformungen von Gleichungssystemen, die in $\mathcal{L}_\Sigma$, aber nicht in jeder stetigen Kleene'schen Algebra erlaubt sind, beruhen auf folgenden Eigenschaften von $\mathcal{L}_\Sigma$:

Proposition 6.25 In der Algebra $\mathcal{L}_\Sigma$ aller Sprachen über Σ gelten folgende Aussagen:

(i) $\forall x\ (x \leq 1 \ \rightarrow\ x = 0 \vee x = 1)$.

(ii) $\forall x \forall y\ (1 \leq x + y \ \rightarrow\ 1 \leq x \vee 1 \leq y)$

(iii) $\forall x \forall y\ (1 \leq x \cdot y \ \rightarrow\ 1 \leq x \wedge 1 \leq y)$.

(iv) $\forall x \forall y\ (x \cdot y = 1 \ \rightarrow\ x = 1 \wedge y = 1)$.

(v) $\forall x \forall y\ (x \cdot y = 0 \ \rightarrow\ x = 0 \vee y = 0)$.

(vi) $\forall x\ (x^* = 1 \ \rightarrow\ x = 0 \vee x = 1)$.

Außerdem gilt für alle "Atome" $a \in \Sigma$:

(i) $\forall x\ (x \leq a \ \rightarrow\ x = 0 \vee x = a)$.

(ii) $\forall x \forall y\ (a \leq x + y \ \rightarrow\ a \leq x \vee a \leq y)$

(iii) $\forall x \forall y\ (a \leq x \cdot y \ \rightarrow\ (1 \leq x \wedge a \leq y) \vee (a \leq x \wedge 1 \leq y))$.

(iv) $\forall x \forall y\ (x \cdot y = a \ \rightarrow\ (x = 1 \wedge y = a) \vee (x = a \wedge y = 1))$.

(v) $\forall x\ (a \leq x^* \ \rightarrow\ a \leq x)$.

Dies ist keine vollständige Liste; z.B. gilt für $a_1, \ldots, a_n \in \Sigma$ auch $1 \neq a_i \neq 0$ und

$$\forall x \forall y\ (x \cdot y \leq (a_1 + \ldots + a_n) \rightarrow\ x \leq 1 \vee y \leq 1).$$

An der Interpretation von $0, 1, +, \cdot$ und den $a \in \Sigma$ in $\mathcal{L}_\Sigma$ sieht man, daß die angegebenen Aussagen in $\mathcal{L}_\Sigma$ gelten. Sie gelten aber z.B. nicht in den Relationeninterpretationen $\mathcal{R}_M$. Beachte, daß die Umkehrungen in jeder Kleene'schen Algebra erfüllt sind.

Neben diesen Eigenschaften setzen die folgenden Umformungen regulärer Gleichungssysteme P voraus, daß man einfache Eigenschaften ihrer kleinsten Lösung $(A_1, \ldots, A_m)$ in $\mathcal{L}_\Sigma$ schon an P ablesen kann:

Proposition 6.26 Für jedes reguläre Gleichungssystem

$$P = \left\{ \begin{array}{ccc} x_1 & = & r_1(x_1, \ldots, x_m) \\ \vdots & \vdots & \vdots \\ x_m & = & r_m(x_1, \ldots, x_m) \end{array} \right.$$

über Σ ist feststellbar, für welche Komponenten A_i der kleinsten Lösung $(A_1, \ldots, A_m) \in (2^{\Sigma^*})^m$ von P in $\mathcal{L}_\Sigma$ die Aussagen $A_i = \emptyset$, $\varepsilon \in A_i$ oder $a \in A_i$ gelten.

Beweis: Man berechnet die Wahrheitswerte der Aussagen "$\varepsilon \in A_i$", "$A_i = \emptyset$" und "$a \in A_i$", ohne die (i.a. unendlichen) Mengen A_i zu berechnen. $(A_1, \ldots, A_m)$ ist der kleinste Fixpunkt von F_P,

$$F_P(B_1, \ldots, B_m) := (r_1^{\mathcal{L}_\Sigma}(B_1, \ldots, B_m), \ldots, r_m^{\mathcal{L}_\Sigma}(B_1, \ldots, B_m)),$$

nämlich die Vereinigung der Stadien $(A_1^n, \ldots, A_m^n) := F_P^n(\emptyset, \ldots, \emptyset)$. Wir behandeln die Aussagen der Form $\varepsilon \in A_i$.

Für jeden regulären Ausdruck $r(x_1, \ldots, x_m)$ und alle $B_1, \ldots, B_m \subseteq \Sigma^*$ können wir den Wahrheitswert von $\varepsilon \in r^{\mathcal{L}_\Sigma}(B_1, \ldots, B_m)$ auf die Werte von $\varepsilon \in B_1, \ldots, \varepsilon \in B_m$ zurückführen, ohne weitere Eigenschaften der B_i zu verwenden. Es genügt, im Halbring

$$\mathcal{B} = (B, +^{\mathcal{B}}, 0^{\mathcal{B}}, \cdot^{\mathcal{B}}, 1^{\mathcal{B}}) := (\{\mathit{wahr}, \mathit{falsch}\}, \vee, \mathit{falsch}, \wedge, \mathit{wahr}).$$

der Wahrheitswerte zu rechnen, mit den üblichen Bedeutungen von $\vee$ ('oder') und $\wedge$ ('und'). Beachte, daß $\vee$ und $\wedge$ bezüglich der durch *falsch* $\leq_{\text{bool}}$ *wahr* gegebenen partiellen Ordnung $\leq_{\text{bool}}$ auf $\mathcal{B}$ monoton sind. Die Bestimmung des Wahrheitswerts von $\varepsilon \in r^{\mathcal{L}_\Sigma}(B_1, \ldots, B_m)$ übersetzen wir in folgende Berechung von $(1 \leq r)^{\mathcal{B}}(b_1, \ldots, b_m)$ durch Induktion über den Aufbau von r, wobei b_i der Wahrheitswert von $\varepsilon \in B_i$ ist:

$$\begin{aligned}
(1 \leq 0)^{\mathcal{B}}(b_1, \ldots, b_m) &= \mathit{falsch}\\
(1 \leq 1)^{\mathcal{B}}(b_1, \ldots, b_m) &= \mathit{wahr}\\
(1 \leq a)^{\mathcal{B}}(b_1, \ldots, b_m) &= \mathit{falsch}\\
(1 \leq x_i)^{\mathcal{B}}(b_1, \ldots, b_m) &= b_i\\
(1 \leq (r+s))^{\mathcal{B}}(b_1, \ldots, b_m) &= (1 \leq r)^{\mathcal{B}}(b_1, \ldots, b_m) \vee (1 \leq s)^{\mathcal{B}}(b_1, \ldots, b_m)\\
(1 \leq (r \cdot s))^{\mathcal{B}}(b_1, \ldots, b_m) &= (1 \leq r)^{\mathcal{B}}(b_1, \ldots, b_m) \wedge (1 \leq s)^{\mathcal{B}}(b_1, \ldots, b_m)\\
(1 \leq r^*)^{\mathcal{B}}(b_1, \ldots, b_m) &= \mathit{wahr}.
\end{aligned}$$

Sei nun b_i^n der Wahrheitswert von $\varepsilon \in A_i^n$. Wegen $A_i^0 = \emptyset$ ist $b_i^0 = \mathit{falsch}$, und der Wert $b_i^{n+1} = (1 \leq r_i)^{\mathcal{B}}(b_1^n, \ldots, b_m^n)$ kann aus $(b_1^n, \ldots, b_m^n)$ nach den obigen Regeln berechnet werden. Da hierbei nur die $\leq_{\text{bool}}$-monotonen Operationen $\vee$ und $\wedge$ benutzt werden, ist $b_i^n \leq_{\text{bool}} b_i^{n+1} \leq_{\text{bool}} \mathit{wahr}$. Es gibt daher ein $n \leq m$ mit $(b_1^n, \ldots, b_m^n) = (b_1^{n+k}, \ldots, b_m^{n+k})$ für alle k. Das sind die gesuchten Wahrheitswerte der Aussagen $\varepsilon \in A_i$:

$$\begin{aligned}
b_i^n = \mathit{wahr} &\implies \varepsilon \in A_i^n \subseteq \bigcup\{\, A_i^k \mid k \in \mathbb{N} \,\} = A_i,\\
b_i^n = \mathit{falsch} &\implies \forall k\ b_i^k = \mathit{falsch} \implies \varepsilon \notin \bigcup\{\, A_i^k \mid k \in \mathbb{N} \,\} = A_i.
\end{aligned}$$

Die Behauptung für $\emptyset \neq A_i$ und $a \in A_i$ geht analog über die Berechung der Wahrheitswerte von $\emptyset \neq r_i^{\mathcal{L}_\Sigma}(B_1, \ldots, B_m)$ und $a \in r_i^{\mathcal{L}_\Sigma}(B_1, \ldots, B_m)$. Dabei werden wieder nur die $\leq_{\text{bool}}$-monotonen Operationen $\wedge$ und $\vee$ verwendet; neben den Werten der Aussagen $\emptyset \neq B_i$ bzw. $a \in B_i$ braucht man für $a \in r_i^{\mathcal{L}_\Sigma}(B_1, \ldots, B_m)$ wegen

$$a \in B \cdot C \iff (a \in B \wedge \varepsilon \in C) \vee (\varepsilon \in B \wedge a \in C)$$

allerdings auch die Werte der Aussagen $\varepsilon \in B_i$. □

Allgemeiner kann man für jedes $w \in \Sigma^*$ feststellen, ob die Aussage $w \in A_i$ für die kleinste Lösung $(A_1, \ldots, A_m)$ von P gilt, wozu man die Monotonie aller Aussagen "$v \in A_j$" für $1 \leq j \leq m$ und alle Teilworte v von w benutzt. (Das wird in Satz 9.7 gezeigt.)

Durch die im Folgenden dargestellten äquivalenten Umformungen kontextfreier Grammatiken kann man Eigenschaften einer Grammatik G, die bei der Syntaxanalyse störend sind, umgehen. Statt mit G betreibt man dann die Analyse mit der umgeformten Grammatik G'. (Da aber die Ausdrücke w von G und G' in unterschiedlicher Form beschrieben werden, muß man in manchen Anwendungen die mit G' gefundenen Analysen allerdings in Analysen bezüglich G rückübersetzen.)

Beseitigung von Löschregeln

Eine LÖSCH- oder ε-REGEL ist eine Produktion der Form $\alpha \to \varepsilon$.

Durch Löschregeln wird die Formulierung von Grammatiken übersichtlicher und kürzer. Sie haben aber den Nachteil, daß bei der Erzeugung $S \overset{*}{\Longrightarrow}_G \alpha \overset{*}{\Longrightarrow}_G w$ kurzer Ausdrücke w sehr große Ausdrucksformen α auftreten können, die dann weitgehend gelöscht werden. Die Analyse muß solche Ableitungen im wesentlichen rekonstruieren und wird entsprechend aufwendig.

Lemma 6.27 Zu jeder kontextfreien Grammatik G gibt es eine kontextfreie Grammatik G' ohne ε-Regeln, so daß $L(G') = L(G) - \{\varepsilon\}$.

Beweis: Wir zeigen allgemeiner: zu jedem regulären Gleichungssystem

$$P = \left\{ \begin{array}{ccc} x_1 & = & r_1(x_1, \ldots, x_m) \\ \vdots & \vdots & \vdots \\ x_m & = & r_m(x_1, \ldots, x_m) \end{array} \right.$$

kann man ein reguläres System $(P - 1) = \{\, x_i = (r_i - 1) \mid 1 \leq i \leq m \,\}$ angeben, so daß $(A_1 - \{\varepsilon\}, \ldots, A_m - \{\varepsilon\})$ die kleinste Lösung von $(P - 1)$ in $\mathcal{L}_\Sigma$ ist, wenn $(A_1, \ldots, A_m)$ die kleinste Lösung von P ist.

Sei $(A_1, \ldots, A_m)$ die kleinste Lösung von P in $\mathcal{L}_\Sigma$. Definiere zu $r(x_1, \ldots, x_m)$ einen regulären Ausdruck $(r - 1)(x_1, \ldots, x_m)$ wie folgt:

$$\begin{array}{lcl} (0 - 1) & := & 0 \\ (1 - 1) & := & 0 \\ (a - 1) & := & a \\ (x_i - 1) & := & x_i \\ ((r + s) - 1) & := & (r - 1) + (s - 1) \\ (r^* - 1) & := & (r - 1) \cdot (r - 1)^* \end{array}$$

$$((r\cdot s)-1) \;:=\; \begin{cases} ((r-1)+1)\cdot(s-1), \\ \qquad \text{falls } \varepsilon\in r^{\mathcal{L}_\Sigma}(A_1,\dots,A_m)-s^{\mathcal{L}_\Sigma}(A_1,\dots,A_m), \\ ((r-1)+1)\cdot(s-1)+(r-1)\cdot((s-1)+1), \\ \qquad \text{falls } \varepsilon\in r^{\mathcal{L}_\Sigma}(A_1,\dots,A_m)\cap s^{\mathcal{L}_\Sigma}(A_1,\dots,A_m), \\ (r-1)\cdot((s-1)+1), \\ \qquad \text{falls } \varepsilon\in s^{\mathcal{L}_\Sigma}(A_1,\dots,A_m)-r^{\mathcal{L}_\Sigma}(A_1,\dots,A_m), \\ (r-1)\cdot(s-1), \qquad \text{sonst} \end{cases}$$

Beachte, daß $(r-1)$ allein aus P bestimmt werden kann, da nach Lemma 6.26 schon an P feststellbar ist, ob $\varepsilon\in A_i$ und damit auch, ob $\varepsilon\in r^{\mathcal{L}_\Sigma}(A_1,\dots,A_m)$ ist. Durch Induktion über r sieht man:

$$r^{\mathcal{L}_\Sigma}(A_1,\dots,A_m)-\{\varepsilon\}=(r-1)^{\mathcal{L}_\Sigma}(A_1-\{\varepsilon\},\dots,A_m-\{\varepsilon\}). \tag{36}$$

Mit der Definition von $(P-1)$ folgt aus (36), daß $(A_1-\{\varepsilon\},\dots,A_m-\{\varepsilon\})$ eine Lösung von $(P-1)$ ist. Sei $(B_1,\dots,B_m)$ irgend eine Lösung von $(P-1)$ in $\mathcal{L}_\Sigma$. Wir zeigen für die Stadien $(A_1^n,\dots,A_m^n)=F_P^n(\emptyset,\dots,\emptyset)$ der Berechnung von $(A_1,\dots,A_m)$:

Beh. Für alle $n\in I\!N$ und alle $r(x_1,\dots,x_m)$ ist

$$r^{\mathcal{L}_\Sigma}(A_1^n,\dots,A_m^n)-\{\varepsilon\}\;\subseteq\;(r-1)^{\mathcal{L}_\Sigma}(B_1,\dots,B_m). \tag{37}$$

Mit $r\equiv x_i$ folgt daraus $A_i^n-\{\varepsilon\}\subseteq B_i$ für alle n, und das ergibt $A_i-\{\varepsilon\}\subseteq B_i$.

Bew: durch Induktion über n.

Fall x_i: Offenbar ist $x_i^{\mathcal{L}_\Sigma}(A_1^0,\dots,A_m^0)-\{\varepsilon\}=\emptyset\subseteq B_i=(x_i-1)^{\mathcal{L}_\Sigma}(B_1,\dots,B_m)$. Im anderen Fall ist nach Induktion

$$\begin{aligned} x_i^{\mathcal{L}_\Sigma}(A_1^{n+1},\dots,A_m^{n+1})-\{\varepsilon\} &= A_i^{n+1}-\{\varepsilon\} \\ &= r_i^{\mathcal{L}_\Sigma}(A_1^n,\dots,A_m^n)-\{\varepsilon\} \\ &\subseteq (r_i-1)^{\mathcal{L}_\Sigma}(B_1,\dots,B_m)=(x_i-1)^{\mathcal{L}_\Sigma}(B_1,\dots,B_m). \end{aligned}$$

Fall $(r\cdot s)$: Wir betrachten den Fall, wo $\varepsilon\in r^{\mathcal{L}_\Sigma}(A_1,\dots,A_m)-s^{\mathcal{L}_\Sigma}(A_1,\dots,A_m)$ ist.

$$\begin{aligned} &(r\cdot s)^{\mathcal{L}_\Sigma}(A_1^n,\dots,A_m^n)-\{\varepsilon\} \\ =\;& r^{\mathcal{L}_\Sigma}(A_1^n,\dots,A_m^n)\cdot(s^{\mathcal{L}_\Sigma}(A_1^n,\dots,A_m^n)-\{\varepsilon\}) \\ \subseteq\;& (((r^{\mathcal{L}_\Sigma}(A_1^n,\dots,A_m^n)-\{\varepsilon\})\cup\{\varepsilon\})\cdot(s-1)^{\mathcal{L}_\Sigma}(B_1,\dots,B_m) \\ \subseteq\;& ((r-1)+1)^{\mathcal{L}_\Sigma}(B_1,\dots,B_m)\cdot(s-1)^{\mathcal{L}_\Sigma}(B_1,\dots,B_m) \\ =\;& ((r\cdot s)-1)^{\mathcal{L}_\Sigma}(B_1,\dots,B_m). \end{aligned}$$

Die übrigen Fälle bleiben dem Leser überlassen. □

Korollar 6.28 Zu jeder kontextfreien Grammatik G mit $\varepsilon \notin L(G)$ gibt es eine äquivalente kontextfreie Grammatik $\tilde{G} = (\tilde{N}, \Sigma, \tilde{P}, \tilde{S})$ ohne ε-Regeln. Zu jeder kontextfreien Grammatik G mit $\varepsilon \in L(G)$ gibt es eine äquivalente kontextfreie Grammatik $\tilde{G}$, in der $\tilde{S} \to \varepsilon$ die einzige ε-Regel ist und $\tilde{S}$ auf keiner rechten Regelseite vorkommt.

Beweis: Betrachte G als reguläres Gleichungssystem P. Nach Lemma 6.27 gibt es ein reguläres System $(P-1)$, dessen kleinste Lösung das Tupel $(A_1 - \{\varepsilon\}, \ldots, A_m - \{\varepsilon\})$ ist, wenn $(A_1, \ldots, A_m)$ die kleinste Lösung von P ist. Falls $\varepsilon \notin L(G)$, sei $\tilde{G}$ die dem System $(P-1)$ nach Ausmultiplizieren der rechten Seiten entsprechende Grammatik –diese hat keine ε-Regeln–, sonst die Grammatik zu $(P-1)$ erweitert um $x_0 = x_1 + 1$ mit neuer Hauptvariable x_0. □

Beseitigung überflüssiger Symbole

Ein Symbol $X \in N \cup \Sigma$ einer kontextfreien Grammatik $G = (N, \Sigma, P, S)$ heißt ÜBERFLÜSSIG in G, falls es keine Ableitungen

$$S \overset{*}{\underset{G}{\Longrightarrow}} uXw \quad \text{und} \quad X \overset{*}{\underset{G}{\Longrightarrow}} v \quad \text{mit} \quad uvw \in \Sigma^*$$

gibt. G heißt CHOMSKY-REDUZIERT, wenn G keine überflüssigen Symbole hat.

Proposition 6.29 Zu jeder kontextfreien Grammatik G mit $L(G) \neq \emptyset$ gibt es eine äquivalente kontextfreie Grammatik G' ohne überflüssige Symbole.

Beweis: Nach Lemma 6.22 können wir annehmen, daß G einem normierten Gleichungssystem $P = \{\, x_i = r_i \mid 1 \leq i \leq m \,\}$ entspricht. Mit 6.26 kann man feststellen, welche der Komponenten A_i der kleinsten Lösung $(A_1, \ldots, A_m)$ von P in $\mathcal{L}_\Sigma$ leer sind; ersetze die entsprechenden r_i durch 0 und beseitige mit 6.20 die scheinbaren Rekursionsvariablen x_i. Durch Vereinfachen der rechten Seiten werden alle Summanden, die einen Faktor 0 enthalten, beseitigt. (Es entstehen dadurch keine neuen Gleichungen $x_j = 0$, da sich die erhalten gebliebenen Komponenten der kleinsten Lösung nicht ändern. Wegen $L(G) \neq \emptyset$ verschwindet die Hauptgleichung nicht.) Gleichungen $x_j = r_j$, die von x_1 nicht erreichbar sind, werden weggelassen. Nach Lemma 6.20 ist das entstandene System P' in $\mathcal{L}_\Sigma$ äquivalent zu P. Es entspricht einer kontextfreien Grammatik G' der angegebenen Form, wobei N' die Rekursionsvariablen und Σ' die Konstanten von P' sind. □

Beispiel 6.30 Sei S die Hauptvariable im System

$$P = \left\{ \begin{array}{lcllcl} S & = & aC + A, & \quad A & = & AB, \\ B & = & AbS + ab, & \quad C & = & c + SC. \end{array} \right.$$

In der kleinsten Lösung $(L(S), L(A), L(B), L(C))$ ist $L(A) = \emptyset = L(B)$, weshalb man die die Gleichungen $A = AB$ und $B = AbS$ durch $A = 0$ und $B = 0$ ersetzt und dann die Rekursionsvariable A und B beseitigt. Das führt zu

$$P' = \begin{cases} S &= aC, \\ B &= ab \qquad C \;=\; c + SC. \end{cases}$$

Die von S nicht erreichbare Gleichung $B = ab$ wird weggelassen, und man hat eine Grammatik ohne überflüssige Symbole.

Beseitigung von Kettenregeln

Eine KETTENREGEL ist eine Regel der Form $A \to B$ mit $A, B \in N$.

Kettenregeln sind insbesondere dann störend, wenn sie Variablenzyklen bilden, also Ableitungen der Form $A \overset{+}{\underset{G}{\Longrightarrow}} B \overset{*}{\underset{G}{\Longrightarrow}} A$ erlauben, die zu Schleifen bei der Syntaxanalyse führen. Außerdem beschreiben Kettenregeln keine Konstruktionen von Ausdrücken aus (echten) Teilausdrücken, sondern lediglich "Umkategorisierungen".

Lemma 6.31 Zu jeder kontextfreien Grammatik G mit $\varepsilon \notin L(G)$ gibt es eine äquivalente kontextfreie Grammatik G' ohne ε- und ohne Kettenregeln.

Beweis: Betrachte G als reguläres System $P = \{\, x_i = r_i \mid 1 \leq i \leq m \,\}$. Eine Kettenregel $x_i \to x_j$ entspricht dann einem Summanden x_j in r_i. Das nach Lemma 6.27 aus P gebildete System $(P - 1)$ enthält (nach Ausmultiplizieren der rechten Seiten) keine Summanden 1 und ist wegen $\varepsilon \notin L(G)$ in $\mathcal{L}_\Sigma$ äquivalent zu P. Wenn es Kettenregeln, also Variable als Summanden, enthält, beseitigt man diese wie im Beweis von Lemma 6.22. Dabei werden keine Summanden 1 eingeführt. □

Definition 6.32 Eine kontextfreie Grammatik $G = (N, \Sigma, P, S)$ ist in CHOMSKY-NORMALFORM, wenn jede Regel die Form

(i) $A \to BC$, mit $A, B, C \in N$, oder

(ii) $A \to a$, mit $A \in N,\ a \in \Sigma$, oder

(iii) $S \to \varepsilon$ hat, wobei in diesem Fall keine rechte Regelseite das Symbol S enthält.

Satz 6.33 (Chomsky 1957) Zu jeder kontextfreien Grammatik $G = (N, \Sigma, P, S)$ gibt es eine äquivalente kontextfreie Grammatik G' in Chomsky-Normalform.

Beweis: Man betrachte G als reguläres Gleichungssystem $P = \{\, x_i = r_i \mid 1 \leq i \leq m \,\}$ und bilde nach 6.27 zunächst das System $(P - 1)$ für $L(G) - \{\varepsilon\}$, wobei man die rechten Seiten ausmultipliziert. Nach 6.28 und 6.31 kann man annehmen, daß es keine Summanden 1 oder x_i enthält. Durch Einführen neuer Rekursionsvariablen nach Lemma 6.20 kann man alle Summanden auf die gewünschte Form bringen. Falls $\varepsilon \in L(G)$ ist, ergänzt man abschließend eine Gleichung $x_0 = x_1 + 1$ und erklärt x_0 zur Hauptvariablen. □

Linksfaktorisierung

Während der Syntaxanalyse muß man immer wieder zwischen verschiedenen Möglichkeiten wählen oder diese nacheinander untersuchen, z.B. bei der Frage, welche Grammatikregel angewendet werden soll. Es kann nun sein, daß diese Alternativen ein teilweise gleiches Vorgehen erlauben und erst in späteren Schritten verschieden vorgegangen werden muß. In solchen Fällen ist es sinnvoll, zuerst den gemeinsamen Teil durchzuführen und sich erst dann zwischen den Alternativen zu entscheiden. Dadurch wird vermieden, dieselbe Arbeit mehrfach zu tun; außerdem kann man zum späteren Zeitpunkt oft besser zwischen den Alternative wählen, da man in der Zwischenzeit mehr Information hat, z.B. weil die Eingabe weiter gelesen wurde.

Lemma 6.34 (Linksfaktorisierung) Seien

$$A \to \alpha_1\beta_{1,1},\ \ldots,\ A \to \alpha_1\beta_{1,k_1}, \quad \ldots \quad A \to \alpha_m\beta_{m,1},\ \ldots,\ A \to \alpha_m\beta_{m,k_m}$$

alle A-Regeln der kontextfreien Grammatik $G = (N, \Sigma, P, S)$. Seien $B_1, \ldots, B_m$ neue Nonterminalsymbole und $G' = (N \cup \{B_1, \ldots, B_m\}, \Sigma, P', S)$ die Grammatik, deren Regeln aus denen von P entstehen, indem man die A-Regeln durch die Regeln

$$\begin{array}{ccc} A \to \alpha_1 B_1, & \ldots, & A \to \alpha_m B_m \\ B_1 \to \beta_{1,1}, & \ldots, & B_1 \to \beta_{1,k_1} \\ \vdots & \vdots & \vdots \\ B_m \to \beta_{m,1}, & \ldots, & B_m \to \beta_{m,k_m} \end{array}$$

ersetzt. Dann ist $L(G') = L(G)$.

Beweis: Betrachtet man G als Gleichungssystem, so ist die Gleichung

$$A = \alpha_1\beta_{1,1} + \ldots + \alpha_1\beta_{1,k_1} + \ldots + \alpha_m\beta_{m,1} + \ldots + \alpha_m\beta_{m,k_m}$$

nach den in Kleene'schen Algebren gültigen Distributivregeln durch

$$A = \alpha_1(\beta_{1,1} + \ldots + \beta_{1,k_1}) + \ldots + \alpha_m(\beta_{m,1} + \ldots + \beta_{m,k_m})$$

ersetzbar. Hierin kann man die Ausdrücke $(\beta_{i,1} + \ldots + \beta_{i,k_i})$ durch B_i und neue Hilfsgleichungen $B_i = (\beta_{i,1} + \ldots + \beta_{i,k_i})$ ersetzen, was die Grammatik mit den angegebenen Regeln ergibt. Nach Lemma 6.20 ändert sich die kleinste Lösung bei dieser Umformung des Gleichungssystems nicht. □

Bei Syntaxanalyse mit G' sucht man zuerst ein zu α_i passendes Präfix der Eingabe und muß sich erst und nur dann, wenn das gelungen ist, zwischen den Alternativen $\beta_{i,1}, \ldots, \beta_{i,k_i}$ entscheiden. Bei der Analyse mit G entscheidet man sich zuerst für eine der Alternativen $\alpha_i\beta_{i,1}, \ldots, \alpha_i\beta_{i,k_i}$; falls es dazu kein passendes Präfix der Eingabe gibt, muß man eine andere Alternative untersuchen und dazu wiederholen, was man bei der Suche nach einem zu α_i passenden Präfix schon getan hat.

Zum Beispiel wird man die Alternativen von $S = \mathit{if}\ E\ \mathit{then}\ S \mid \mathit{if}\ E\ \mathit{then}\ S\ \mathit{else}\ S \mid a$ zu $S = \mathit{if}\ E\ \mathit{then}\ S\ B \mid a, B = \mathit{else}\ S \mid \epsilon$ zusammenfassen.

Beseitigung von Linksrekursion

Ein Nonterminal $A \in N$ ist LINKSREKURSIV in $G = (N, \Sigma, P, S)$, falls es eine Ableitung $A \Longrightarrow_G^+ A\alpha$ für ein $\alpha \in (N \cup \Sigma)^*$ gibt.

Bei der Top-Down-Analyse (vgl. Abschnitt 9.2) geht man von der Erwartung aus, daß das Eingabewort $w \in \Sigma^*$ von einer bestimmten Kategorie A sei. Dann versucht man, durch Anwendung der Produktionsregeln aus A einen Ausdruck zu erzeugen und mit der Eingabe zu vergleichen. Ist aber A eine linksrekursive Kategorie, so führt dieses Vorgehen dazu, daß man immer längere Ableitungen $A \Longrightarrow_G^+ A\alpha \Longrightarrow_G^+ A\alpha\alpha \ldots$ aufbaut, ohne ein Terminalsymbol zu erzeugen, das man mit der Eingabe w vergleichen könnte. Linksrekursive Kategorien machen also die Top-Down-Syntaxanalyse unmöglich.

Man möchte daher eine Grammatik so umformen können, daß jede Phrasenform $\alpha \in (\Sigma \cup N)^*$ mit einem Buchstaben $a \in \Sigma$ beginnt und daher direkt mit einem Eingabewort $w \in \Sigma^+$ verglichen werden kann.

Die Konstruktion wird durchschaubarer, wenn wir mit in der nach Lemma 6.7 stetigen Kleene'schen Algebra der $n \times n$-Matrizen über $\mathcal{L}_\Sigma$ arbeiten.

Satz 6.35 (Ersetzung von Links- durch Rechtsrekursion) Die Ausdrücke $r_i(x_1, \ldots, x_m)$ des regulären Gleichungssystems

$$\begin{array}{rcl} x_1 & = & r_1(x_1, \ldots, x_m) \\ \vdots & \vdots & \vdots \\ x_m & = & r_m(x_1, \ldots, x_m) \end{array} \tag{38}$$

seien Summen von Produkten aus Buchstaben $a \in \Sigma$ und Variablen $x_1, \ldots, x_m$, ohne Summanden 1 und x_i. Dann gibt es ein reguläres Gleichungssystem

$$\begin{array}{rclcl} x_1 & = & a_{1,1} \cdot s_{1,1}(x_1, \ldots, x_k) & + \ldots + & a_{1,l_1} \cdot s_{1,l_1}(x_1, \ldots, x_k) \\ \vdots & \vdots & & \vdots & \\ x_k & = & a_{k,1} \cdot s_{k,1}(x_1, \ldots, x_k) & + \ldots + & a_{k,l_k} \cdot s_{k,l_k}(x_1, \ldots, x_k), \end{array} \tag{39}$$

mit $m \leq k \leq m(m+1)$, so daß

(i) für alle $i, j \leq k$ ist $a_{i,j} \in \Sigma$ und $s_{i,j} \in (\Sigma \cup \{x_1, \ldots, x_k\})^*$, und

(ii) In $\mathcal{L}_\Sigma$ stimmen die kleinsten Lösungen von (38) und (39) in den ersten m Komponenten überein.

Beweis: Wir bringen das System (38) durch Ausklammern der führenden Variablen und Konstanten auf die Form

$$\begin{array}{rcrlll} x_1 & = & & x_1 \cdot r_{1,1}(x_1,\ldots,x_m) & +\ldots+ & x_m \cdot r_{m,1}(x_1,\ldots,x_m) \\ & & + & a_1 \cdot t_{1,1}(x_1,\ldots,x_m) & +\ldots+ & a_n \cdot t_{n,1}(x_1,\ldots,x_m) \\ \vdots & \vdots & & & \vdots & \\ x_m & = & & x_1 \cdot r_{1,m}(x_1,\ldots,x_m) & +\ldots+ & x_m \cdot r_{m,m}(x_1,\ldots,x_m) \\ & & + & a_1 \cdot t_{1,m}(x_1,\ldots,x_m) & +\ldots+ & a_n \cdot t_{n,m}(x_1,\ldots,x_m). \end{array} \tag{40}$$

Nach der Voraussetzung über die r_i tritt in keinem $r_{i,j}$ ein Summand 1 auf. Dieses System (40) läßt sich mit Vektoren und Matrizen in der Form

$$(x_1,\ldots,x_m) \;=\; (x_1,\ldots,x_m)\cdot R(x_1,\ldots,x_m) \;+\; (t_1,\ldots,t_m) \tag{41}$$

schreiben, mit

$$R = \begin{pmatrix} r_{1,1} & \ldots & r_{1,m} \\ \vdots & \ddots & \vdots \\ r_{m,1} & \ldots & r_{m,m} \end{pmatrix} \qquad \text{und} \qquad t_j = \Sigma_{i=1}^{n} a_i \cdot t_{i,j}.$$

Sei $\vec{x} := (x_1,\ldots,x_m)$ und $\vec{t} := (t_1,\ldots,t_m)$. Die kleinste Lösung der *links*rekursiven Gleichung

$$\vec{x} = \vec{x}\cdot R \;+\; \vec{t} \tag{42}$$

ist gerade die kleinste Lösung von

$$\vec{x} = \vec{t}\cdot R^*. \tag{43}$$

Wir können uns nämlich (42) und (43) als Matrixgleichungen vorstellen, indem wir die Vektoren $\vec{x}$ und $\vec{t}$ mit 0-Einträgen zu $m \times m$-Matrizen auffüllen, und nach Lemma 6.18 erfüllt jede stetige Kleene'sche Algebra, insbesondere $\mathcal{M}_{m,m}(\mathcal{L}_\Sigma)$, die Formeln

$$\mu x(x\cdot r + t) = \mu x(t \cdot r^*) = \mu x(t\cdot r^+ + t),$$

auch wenn x frei in r und t vorkommt. Nun ist R^* aus (43) aber die kleinste Lösung der *rechts*rekursiven Gleichung $Y = R\cdot Y + 1$ bezüglich einer unbekannten Matrix

$$Y = \begin{pmatrix} y_{1,1} & \cdots & y_{1,m} \\ \vdots & \ddots & \vdots \\ y_{m,1} & \ldots & y_{m,m} \end{pmatrix}$$

aus neuen Variablen $y_{i,j}$. Um die 1 in der Rekursionsbedingung für R^* und damit das Auftreten von Summanden 1 im Ergebnis zu vermeiden, ersetzen wir (43) durch die äquivalente Bedingung

$$\vec{x} \;=\; \vec{t}\cdot R^+ + \vec{t},$$

wobei R^+ in $\mathcal{M}_{m,m}(\mathcal{L}_\Sigma)$ die kleinste Lösung der rechtsrekursiven Gleichung

$$Y = R\cdot Y \;+\; R$$

ist. Wir betrachten also statt (41) das simultane System

$$\begin{array}{rcl} \vec{x} & = & \vec{t} \cdot Y + \vec{t} \\ Y & = & R \cdot Y + R \end{array} \tag{44}$$

als Gleichungssystem in den Variablen $x_1, \ldots, x_m, y_{1,1}, \ldots, y_{m,m}$. Die kleinsten Lösungen von (41) und (44) stimmen auf $x_1, \ldots, x_m$ überein.

Offensichtlich kann (44) leicht in ein Gleichungssystem der gewünschten Form (39) gebracht werden: in den Gleichungen

$$\begin{array}{rcl} x_j & = & (\vec{t} \cdot Y)_j + t_j = \Sigma_{k=1}^m t_k \cdot y_{k,j} + t_j \\ & = & \Sigma_{k=1}^m (\Sigma_{i=1}^n a_i \cdot t_{i,k}) \cdot y_{k,j} + \Sigma_{i=1}^n a_i \cdot t_{i,j} \end{array} \tag{45}$$

hat jeder Summand einen führenden Faktor $a \in \Sigma$; wenn alle $t_{i,k} = 0$ sind, können wir die Gleichung durch $x_j = a \cdot x_j$ für ein beliebiges $a \in \Sigma$ ersetzen. In den Gleichungen

$$y_{i,j} = \Sigma_{k=1}^m r_{i,k}(x_1, \ldots, x_m) \cdot y_{k,j} + r_{i,j}(x_1, \ldots, x_m) \tag{46}$$

können wir die führenden Variablen unter $x_1, \ldots, x_m$ durch die rechten Seiten der Gleichungen (45) ersetzen. Da kein $r_{i,j}$ einen Summanden 1 hatte, tritt in (46) nach dem Ausmultiplizieren auch kein führendes $y_{k,j}$ und kein Summand 1 auf. Also ist das Ergebnis ein Gleichungssystem der Form (39). □

Korollar 6.36 (Greibach-Normalform) Sei G eine kontextfreie Grammatik über Σ mit $\varepsilon \notin L(G)$. Dann gibt es eine kontextfreie Grammatik G' mit $L(G') = L(G)$, deren Regeln von der Form $A \to a\beta$ sind, wobei a ein Terminalsymbol und β eine Folge von Terminal- und Nonterminalsymbolen ist.

Beweis: Wegen $\varepsilon \notin L(G)$ können wir eine zu G äquivalente kontextfreie Grammatik ohne ε- und Kettenregeln bilden. Das entsprechende reguläre Gleichungssystem (38) erfüllt dann die Voraussetzungen von Satz 6.35. Wir können es also in ein äquivalentes System der Form (39) umwandeln. Das entspricht einer Grammatik der gewünschten Form. □

Beispiel 6.37 Sei G die kontextfreie Grammatik über $\Sigma = \{a, b\}$ und $N = \{A, B, C\}$, die dem System

$$\begin{array}{rcl} A & = & BC + a \\ B & = & Ab + CA \\ C & = & AB + CC \end{array} \tag{47}$$

entspricht. In der Matrixschreibweise ist das gerade die linksrekursive Gleichung

$$(A, B, C) \;=\; (A, B, C) \cdot \begin{pmatrix} 0 & b & B \\ C & 0 & 0 \\ 0 & A & C \end{pmatrix} \;+\; (a, 0, 0). \tag{48}$$

Ersetzt man diese Gleichung durch das rechtsrekursive Gleichungssystem (44) des Beweises, so erhält man

$$\begin{array}{rcl} (A,B,C) & = & (a,0,0)\cdot Y + (a,0,0) \\ Y & = & R\cdot Y + R \end{array} \tag{49}$$

mit

$$R = \begin{pmatrix} 0 & b & B \\ C & 0 & 0 \\ 0 & A & C \end{pmatrix} \qquad \text{und unbekanntem} \quad Y = \begin{pmatrix} Y_{1,1} & \cdots & Y_{1,3} \\ \vdots & \ddots & \vdots \\ Y_{3,1} & \cdots & Y_{3,3} \end{pmatrix}.$$

Ausmultiplizieren führt von (49) zu folgendem Gleichungssystem:

$$\begin{array}{rclrcl} A & = & aY_{1,1}+a & Y_{1,1} & = & bY_{2,1}+BY_{3,1} \\ B & = & aY_{1,2} & Y_{1,2} & = & bY_{2,2}+BY_{3,2}+b \\ C & = & aY_{1,3} & Y_{1,3} & = & bY_{2,3}+BY_{3,3}+B \\ Y_{2,1} & = & CY_{1,1}+C & Y_{3,1} & = & AY_{2,1}+CY_{3,1} \\ Y_{2,2} & = & CY_{1,2} & Y_{3,2} & = & AY_{2,2}+CY_{3,2}+A \\ Y_{2,3} & = & CY_{1,3} & Y_{3,3} & = & AY_{2,3}+CY_{3,3}+C \end{array} \tag{50}$$

Setzen wir jetzt die Gleichungen für A, B, C in die übrigen ein, so erhalten wir ein System der gewünschten Form (39):

$$\begin{array}{rclrcl} A & = & aY_{1,1}+a & Y_{1,1} & = & bY_{2,1}+aY_{1,2}Y_{3,1} \\ B & = & aY_{1,2} & Y_{1,2} & = & bY_{2,2}+aY_{1,2}Y_{3,2}+b \\ C & = & aY_{1,3} & Y_{1,3} & = & bY_{2,3}+aY_{1,2}Y_{3,3}+aY_{1,2} \\ Y_{2,1} & = & aY_{1,3}Y_{1,1}+aY_{1,3} & Y_{3,1} & = & aY_{1,1}Y_{2,1}+aY_{2,1}+aY_{1,3}Y_{3,1} \\ Y_{2,2} & = & aY_{1,3}Y_{1,2} & Y_{3,2} & = & aY_{1,1}Y_{2,2}+aY_{2,2}+aY_{1,3}Y_{3,2}+aY_{1,1}+a \\ Y_{2,3} & = & aY_{1,3}Y_{1,3} & Y_{3,3} & = & aY_{1,1}Y_{2,3}+aY_{2,3}+aY_{1,3}Y_{3,3}+aY_{1,3} \end{array}$$

Zählt man die Summanden als Produktionsregeln, so erhält man eine Grammatik mit 28 Regeln. Das in Hopcroft/Ullman[11] angegebene Verfahren ergibt für dieses Beispiel eine Grammatik mit 119 Regeln.

Aufgaben

Aufgabe 6.1 Man gebe zu der durch das Gleichungssystem

$$P = \left\{ \begin{array}{rclrcl} A & = & BCD & B & = & bC+c \\ C & = & Cc+DD & D & = & aDb+1 \end{array} \right.$$

gegebenen kontextfreien Grammatik G zunächst das System $(P-1)$ nach 6.27 an und dann eine äquivalente kontextfreie Grammatik ohne ε-Regeln und ohne Kettenregeln an.

Aufgabe 6.2 Sei G die folgende kontextfreie Grammatik für Nominalphrasen im Plural (ohne Kasus-, Genus-, Numerusmerkmale):

1: NP → die AP N	4: AP → sehr A
2: NP → AP N	5: AP → NP AV
3: NP → NP und NP	6: AP → A
7: N → Studenten	9: A → leichte
8: N → Aufgaben	10: A → schwierige
9: N → Probleme	11: AV → lösenden

Mit ihr kann man z.B. die Nominalphrase

(*) `die schwierige Aufgaben und sehr leichte Probleme lösenden Studenten`

erzeugen.

(i) Man gebe die Teilgrammatik aus den Regeln 1 bis 6 als Gleichungssystem bzw. als Matrixgleichung an.

(ii) Diese Teilgrammatik nach (i) enthält eine direkte und eine indirekte Linksrekursion. Man konstruiere nach dem Verfahren aus Abschnitt 6.4 eine zu ihr äquivalente Grammatik ohne Linksrekursion.

Aufgabe 6.3 Man beweise Satz 6.14.

Aufgabe 6.4 Zeige durch eine geeignete Wahl von Grundrelationen $\vdash_a \subseteq M \times M$, daß in der Relationenalgebra $\mathcal{R} = (2^{M\times M}, \cup, \emptyset, \circ, 1_M, {}^*, \vdash_a)_{a\in\Sigma}$ die Inklusionen

$$\begin{array}{lcllcl} \vdash_{L_1-L_2} & \supseteq & \vdash_{L_1} - \vdash_{L_2} \qquad & \vdash_{L_1\cap L_2} & \subseteq & \vdash_{L_1} \cap \vdash_{L_2} \\ \vdash_{L_1/L_2} & \subseteq & \vdash_{L_1} / \vdash_{L_2} \qquad & \vdash_{L_1\backslash L_2} & \subseteq & \vdash_{L_1} \backslash \vdash_{L_2}\,. \end{array}$$

aus Proposition 6.15 im allgemeinen echt sind.

Aufgabe 6.5 Ist es entscheidbar, ob zwei Kategorien X, Y einer kontextfreien Grammatik G über Σ dieselbe Wortmenge über Σ definieren, d.h. ob $L(X) = L(Y)$ ist?

7 Erkennung kontextfreier Sprachen

Wie bei Sprachen, die durch reguläre Ausdrücke definiert werden, stellt sich auch bei den gleichungs- oder μ-regulär definierbaren Sprachen die Frage, ob und wie man sie durch Automaten erkennnen kann.

7.1 Rekursive endliche Automaten

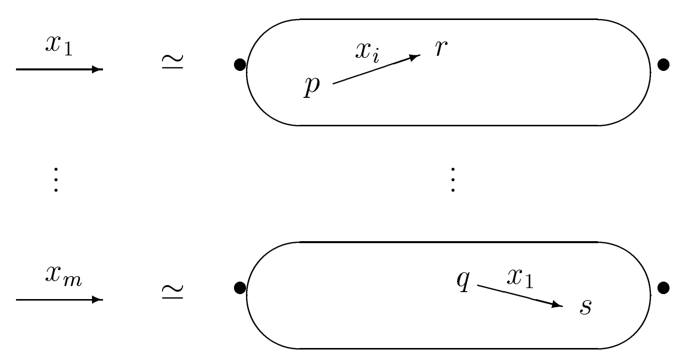
Für gleichungsdefinierte Sprachen liegt nahe, was zu tun ist. Zu jedem regulären Ausdruck $r_j(x_1, \ldots, x_m)$ eines regulären Gleichungssystems

$$\begin{array}{ccc} x_1 & = & r_1(x_1, \ldots, x_m) \\ \vdots & \vdots & \vdots \\ x_m & = & r_m(x_1, \ldots, x_m) \end{array}$$

über Σ gibt es einen endlichen Automaten

$$\mathcal{A}_j = (Q^j, \vdash^j_a, \vdash^j_{x_1}, \ldots, \vdash^j_{x_m}, I^j, F^j)_{a \in \Sigma},$$

der eine Sprache $L(A_j) \subseteq (\Sigma \cup \{x_1, \ldots, x_m\})^*$ akzeptiert. Wir haben also ein System von endlichen Automaten,



die in geeigneter Weise zusammenarbeiten sollten, um die Komponenten der kleinsten Lösung $(L_1, \ldots, L_m)$ des Gleichungssystems zu erkennen.

Bei der Erkennung werden die Grundrelationen $\vdash^j_a$ für $a \in \Sigma$ benutzt, wenn ein Eingabesymbol $a \in \Sigma$ gelesen wird. Eine mit x_i markierte Kante in einem der $\mathcal{A}_j$ darf genau dann passiert werden, wenn der Automat $\mathcal{A}_i$ eine Folge von Eingabesymbolen erkannt hat. Im Zustand p bedeutet ein Schritt $p \vdash_{x_i} r$, den der Gleichung $x_i = r_i(x_1, \ldots, x_m)$ entsprechenden Automaten $\mathcal{A}_i$ AUFZURUFEN, und wenn $\mathcal{A}_i$ einen Endzustand erreicht hat, in den aufrufenden Automaten ZURÜCKZUKEHREN und im Zustand r fortzufahren.

Solche Systeme $(\mathcal{A}_1, \ldots, \mathcal{A}_m)$ sich gegenseitig rekursiv aufrufender endlicher Automaten treten gelegentlich unter dem Namen REKURSIVE ENDLICHE AUTOMATEN beim Übersetzerbau in der Informatik (z.B. Gough[8]) und als REKURSIVE TRANSITIONSNETZWERKE in der Computerlinguistik auf.

Die Verwaltung der rekursiven Aufrufe kann mit einem endlichen Speicher nicht mehr bewältigt werden. Bei jedem rekursiven Aufruf muß der Zustand gespeichert werden, in dem nach Beendigung des Rekursionsaufrufs weiterzuarbeiten ist. (Dies ist die "Rücksprungadresse", wenn man Zustände als Zeilennummern eines Programms interpretiert.) Bei geschachtelten Rekursionsaufrufen wird der erste Aufruf zuletzt abgeschlossen, so daß man seine Rücksprungadresse erst zuletzt wieder braucht; man kann einen STAPEL oder KELLER benutzen, um sich die Rücksprungadressen in der richtigen Reihenfolge zu merken. Da die Schachtelungstiefe der Aufrufe mit der Länge der Eingaben unbeschränkt wachsen kann, kann die Größe des Stapels nicht beschränkt werden; man braucht einen *potentiell unendlichen* Speicher. Um Zustände aus Q zu stapeln, nehmen wir die Folgen aus Q^* als erlaubte Inhalte des Stapels.

In der folgenden Definition fassen wir die Zustandsmengen Q^j und Übergangsrelationen $\vdash_a^j$ der Teilautomaten zu jeweils einer Zustandmenge Q bzw. Relation $\vdash_a$ zusammen, beseitigen die Hilfsrelationen $\vdash_{x_i}^j$ und stellen das Aufrufen und Zurückkehren durch neue Hilfsrelationen $\vdash_{call}$ und $\vdash_{return}$ dar.

Definition 7.1 Ein REKURSIVER ENDLICHER AUTOMAT,

$$\mathcal{A} = (Q, \vdash_a, \vdash_{call}, \vdash_{return}, I, F)_{a \in \Sigma},$$

besteht aus (i) endlichen Mengen Σ und Q, dem Alphabet und den möglichen Zuständen des Arbeitsspeichers, (ii) einer Übergangsrelation $\vdash_a \subseteq Q \times Q$ für jedes $a \in \Sigma$, (iii) Relationen $\vdash_{call} \subseteq Q \times QQ$ und $\vdash_{return} \subseteq Q \times \{\varepsilon\}$, und (iv) Mengen $I \subseteq Q$ von Anfangszuständen und $F \subseteq Q$ von Endzuständen.

Die in $\mathcal{A}$ angegebenen Relationen werden wie folgt zu (unendlichen) Relationen auf $Q^* \times Q^*$ fortgesetzt: für jedes $\gamma \in Q^*$, $p, q, r \in Q$ und $a \in \Sigma$ sei

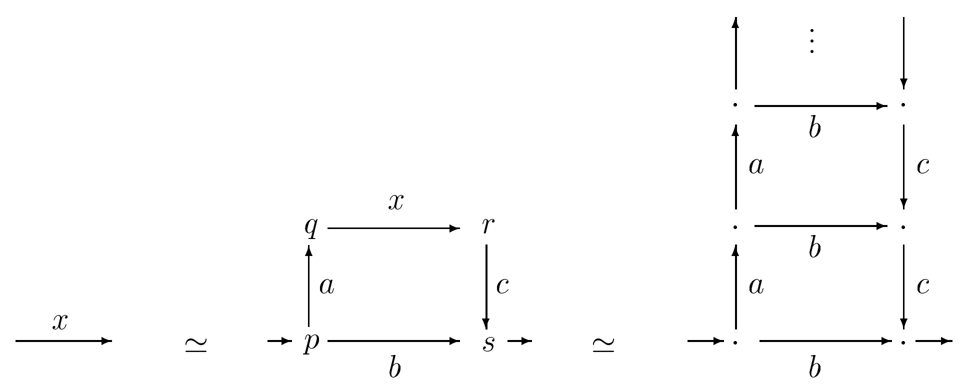
$$\begin{array}{rcl} \gamma p \vdash_a \ \gamma q & :\Longleftrightarrow & p \vdash_a \ q, \\ \gamma p \vdash_{call} \ \gamma r q & :\Longleftrightarrow & p \vdash_{call} \ rq, \\ \gamma p \vdash_{return} \ \gamma & :\Longleftrightarrow & p \vdash_{return} \varepsilon. \end{array}$$

Mit Hilfe des Relationenprodukts erhalten wir hieraus Interpretationen $\vdash_w \subseteq Q^* \times Q^*$ beliebiger Wörter $w \in (\Sigma \cup \{call, return\})^*$. Die VON $\mathcal{A}$ ERKANNTE SPRACHE ist

$$L(\mathcal{A}) := \{ \, erase(w) \mid w \in (\Sigma \cup \{call, return\})^*, \ \exists p \in I, q \in F \ p \ \vdash_w \ q \, \},$$

wobei $erase(w)$ das Wort in Σ^* ist, das man aus w durch Löschen aller *call*'s und *return*'s erhält.

Beispiel 7.2 Die Sprache $L = \{\, a^n b c^n \mid n \in \mathbb{N} \,\}$ ist die kleinste Lösung der Rekursionsbedingung $x = axc + b$. Dieser Gleichung entspricht der sich selbst aufrufende Automat



Durch Beseitigen der x-Übergänge ergibt das den rekursiven endlichen Automaten

$$\mathcal{A} \;=\; (Q, \vdash_a, \vdash_b, \vdash_c, \vdash_{call}, \vdash_{return}, I, F)$$

mit

$$\begin{array}{lllllllll}
Q & = & \{p,q,r,s\}, & I & = & \{p\}, & F & = & \{s\}, \\
\vdash_a & = & \{(p,q)\}, & \vdash_b & = & \{(p,s)\}, & \vdash_c & = & \{(r,s)\}, \\
\vdash_{call} & = & \{(q,rp)\}, & \vdash_{return} & = & \{(s,\varepsilon)\}. & & &
\end{array}$$

Eine das Wort $aabcc$ akzeptierende Berechnung von $\mathcal{A}$ ist

$$p \vdash_a q \vdash_{call} rp \vdash_a rq \vdash_{call} rrp \vdash_b rrs \vdash_{return} rr \vdash_c rs \vdash_{return} r \vdash_c s.$$

Die erkannte Sprache L ist die kleinste Lösung der Gleichung $x = axb + c$ durch eine Sprache über $\Sigma = \{a, b, c\}$, so daß $L(\mathcal{A}) = L(\mu x(axb + c))$. Wir erhalten $\mathcal{A}$, wenn wir aus dem endlichen Automaten $\mathcal{A}_{(axb+c)} = (Q, \vdash_a, \vdash_b, \vdash_c, \vdash_x, I, F)$ mit $\vdash_x = \{(p,r)\}$ (und x als Eingabesymbol) einen rekursiven endlichen Automaten $\mathcal{A} = \mathcal{A}_{\mu x(axb+c)}$ machen, indem wir x-Übergänge durch Rekursionsaufrufe interpretieren.

Der folgende Satz gibt eine allgemeine Automatenkonstruktion μx an, die *Rekursion in* x, die aus einem (ggf. rekursiven) endlichen Automaten $\mathcal{A}_r$ mit $L(\mathcal{A}_r) = L(r)$ einen rekursiven endlichen Automaten $\mathcal{A}_{\mu x.r} = \mu x(\mathcal{A}_r)$ macht, so daß $L(\mathcal{A}_{\mu x.r}) = L(\mu x.r)$.

Wie erwartet lassen sich daher mit rekursiven endlichen Automaten alle gleichungsdefinierbaren Sprachen erkennen. Für den Beweis können wir nach 6.13 statt der Gleichungssysteme die μ-regulären Ausdrücke betrachten. Daß umgekehrt jeder rekursive endliche Automat eine gleichungsdefinierbare Sprache erkennt, folgt in Satz 7.8.

Satz 7.3 Zu jedem μ-regulären Ausdruck r über Σ (ohne freie Variablen) gibt es einen rekursiven endlichen Automaten $\mathcal{A}_r$ über Σ mit $L(\mathcal{A}_r) = L(r)$.

Beweis: Durch Induktion über den Aufbau der μ-regulären Ausdrücke. Dazu muß die Behauptung für Ausdrücke $r(x_1, \dots, x_m)$ mit freien Variablen verallgemeinert werden:

Beh.: Zu jedem μ-regulären Ausdruck $r(x_1, \dots, x_m)$ über Σ gibt es einen rekursiven endlichen Automaten $\mathcal{A}_{r(x_1,\dots,x_m)}$ über $\Sigma \cup \{x_1, \dots, x_m\}$, so daß

$$L(\mathcal{A}_{r(x_1,\dots,x_m)}) = L(r(x_1, \dots, x_m)) \subseteq (\Sigma \cup \{x_1, \dots, x_m\})^*.$$

Wir setzen den Beweis für den Satz 4.5 von Kleene fort. Dort hatten wir induktiv über den Aufbau eines regulären Ausdrucks r einen endlichen Automaten $\mathcal{A}_r$ mit $L(\mathcal{A}_r) = L(r)$ definiert (ohne ε-Übergänge zu verwenden). Dazu wurden Automaten $\mathcal{A}_0, \mathcal{A}_1$ und $\mathcal{A}_a$, für $a \in \Sigma$, mit

$$L(\mathcal{A}_0) = \emptyset, \quad L(\mathcal{A}_1) = \{\varepsilon\}, \quad L(\mathcal{A}_a) = \{a\}$$

angegeben und die regulären Operationen $+, \cdot$ und * durch die Konstruktionen $\mathcal{A}+\mathcal{B}$, $\mathcal{A}\cdot\mathcal{B}$ und $\mathcal{A}^*$ der Parallelschaltung, Hintereinanderschaltung und Rückkopplung interpretiert und gezeigt, daß damit gilt:

$$L(\mathcal{A}+\mathcal{B}) = L(A) \cup L(\mathcal{B}), \quad L(\mathcal{A}\cdot\mathcal{B}) = L(\mathcal{A}) \cdot L(\mathcal{B}), \quad \text{und} \quad L(\mathcal{A}^*) = L(\mathcal{A})^*.$$

Dieselben Konstruktionen kann man auch für rekursive endliche Automaten über $\Sigma \cup \{x_1, \dots, x_m\}$ benutzen, wobei $\mathcal{A}_{x_i}$ wie die $\mathcal{A}_a$ definiert wird; für $\vdash_{call}$ und $\vdash_{return}$ nimmt man bei $\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_a$ und den $\mathcal{A}_{x_i}$ die leere Relation und bei der Parallelschaltung, Hintereinanderschaltung und Rückkopplung die Vereinigungen der jeweiligen Relationen $\vdash_{call}$ und $\vdash_{return}$ der beteiligten Automaten. Für so konstruierte $\mathcal{A}$ über $\Sigma \cup \{x_1, \dots, x_m\}$ sind die durch

$$(X_1, \dots, X_m) \quad \mapsto \quad L(\mathcal{A})[X_1/x_1, \dots, X_m/x_m]$$

definierten Abbildungen von $2^{\Sigma*} \times \dots \times 2^{\Sigma^*}$ nach 2^{Σ^*} stetig, haben also einen kleinsten Fixpunkt.

Zum Nachweis der Behauptung führen wir auch für die nicht-reguläre Operation μx eine Konstruktion von Automaten ein, die Rekursion in x, die aus einem rekursiven endlichen Automaten über $\Sigma \cup \{x, x_1, \dots, x_m\}$ einen rekursiven endlichen Automaten über $\Sigma \cup \{x_1, \dots, x_m\}$ macht; zur Vereinfachung der Schreibweise unterdrücken wir dabei die Variablen $x_1, \dots, x_m$. Aus einem rekursiven endlichen Automaten

$$\mathcal{A} = (Q, \vdash_a, \vdash_x, \vdash_{call}, \vdash_{return}, I, F)_{a\in\Sigma},$$

über $\Sigma \cup \{x\}$, mit $\vdash_x \subseteq Q \times Q$, entsteht DURCH REKURSION VON $\mathcal{A}$ IN x der rekursive endliche Automat

$$\mu x\,(\mathcal{A}) := (Q, \vdash_a, \vdash'_{call}, \vdash'_{return}, I, F)_{a\in\Sigma}$$

über Σ, wobei

$$\begin{aligned} \vdash'_{call} &:= \vdash_{call} \cup \{\, (p, qq_0) \mid p \vdash_x q,\ q_0 \in I \,\}, \\ \vdash'_{return} &:= \vdash_{return} \cup \{\, (p, \varepsilon) \mid p \in F \,\}. \end{aligned}$$

Beh.: Die von $\mu x(\mathcal{A})$ erkannte Sprache ist die kleinste Sprache $X \subseteq \Sigma^*$ mit $X = L(\mathcal{A})[X/x]$:

$$L(\mu x(\mathcal{A})) = \mu X.\ L(\mathcal{A})[X/x]. \tag{51}$$

(Im allgemeinen Fall läuft X über Sprachen im Alphabet von $\mathcal{A}$ ohne den Buchstaben x.) Daraus folgt die eigentliche Behauptung, indem wir zu einem μ-regulären Ausdruck $\mu x.r$ den Automaten $\mathcal{A}_{\mu x.r} := \mu x(\mathcal{A}_r)$ wählen: dann ist mit Induktion

$$L(\mathcal{A}_{\mu x.r}) = \mu X.\ L(\mathcal{A}_r)[X/x] = \mu X.\ L(r)[X/x] = L(\mu x.r).$$

Beh. 1: $\mu X.\, L(\mathcal{A})[X/x] \subseteq L(\mu x(\mathcal{A}))$:

Da nach dem Fixpunktsatz $\mu X.L(\mathcal{A})[X/x] = \bigcup_{n\in\mathbb{N}} L_n(\mathcal{A})$ ist, mit $L_0(\mathcal{A}) := \emptyset$ und $L_{n+1}(\mathcal{A}) := L(\mathcal{A})[L_n(\mathcal{A})/x]$, zeigen wir durch Induktion über n, daß $L_n(\mathcal{A}) \subseteq L(\mu x(\mathcal{A}))$.

$n = 0$: $L_0(\mathcal{A}) = \emptyset \subseteq L(\mu x(\mathcal{A}))$.

$n+1$: Sei $w \in L_{n+1}(\mathcal{A}) = L(\mathcal{A})[L_n(\mathcal{A})/x]$. Es gibt $\alpha \in (\Sigma \cup \{x, \mathit{call}, \mathit{return}\})^*$ und $q_0 \in I, q_e \in F$ mit

$$q_0 \vdash_\alpha q_e \in F \quad \text{und} \quad w \in L(\mathit{erase}(\alpha))[L_n(\mathcal{A})/x].$$

Falls x in α nicht vorkommt, ist $q_0 \vdash'_\alpha q_e \in F$ in $\mu x(\mathcal{A})$ und daher $w = \mathit{erase}(\alpha) \in L(\mu x(\mathcal{A}))$. Angenommen, x kommt - o.B.d.A. nur einmal – in α vor und $\alpha = \alpha_1 \cdot x \cdot \alpha_2$. Dann ist $w = \mathit{erase}(\alpha_1) \cdot v \cdot \mathit{erase}(\alpha_2)$ mit $v \in L_n(\mathcal{A}) \subseteq L(\mu x(\mathcal{A}))$. Also gibt es $\beta \in (\Sigma \cup \{\mathit{call}, \mathit{return}\})^*$ mit $v = \mathit{erase}(\beta)$ und $q_{0,v} \in I, q_{e,v} \in F$ mit $q_{0,v} \vdash'_\beta q_{e,v}$. Sei

$$\tilde{\alpha} := \alpha_1 \cdot \mathit{call} \cdot \beta \cdot \mathit{return} \cdot \alpha_2 \ \in (\Sigma \cup \{\mathit{call}, \mathit{return}\})^*,$$

also $\mathit{erase}(\tilde{\alpha}) = w$. Zu einer Berechnung $q_0 \vdash_{\alpha_1} \gamma p \vdash_x \gamma q \vdash_{\alpha_2} q_e \in F$ von $\mathcal{A}$ ist

$$q_0 \vdash_{\alpha_1} \gamma p \vdash'_{\mathit{call}} \gamma q q_{0,v} \vdash_\beta \gamma q q_{e,v} \vdash'_{\mathit{return}} \gamma q \vdash_{\alpha_2} q_e \in F$$

eine Berechnung von $\mu x(\mathcal{A})$, wegen der neuen Interpretationen von *call* und *return*. Dies zeigt, daß $q_0 \vdash'_{\tilde{\alpha}} q \in F$ und daher $w \in L(\mu x(\mathcal{A}))$. Der Fall mit mehreren Vorkommen von x in α geht entsprechend.

Beh. 2: $L(\mu x(\mathcal{A})) \subseteq \mu X.\ L(\mathcal{A})[X/x]$:

Sei $w \in L(\mu x(\mathcal{A}))$ und $w = \mathit{erase}(\alpha)$ mit $q_0 \vdash'_\alpha q_e$ für geeignete $q_0 \in I, q_e \in F$. Beachte, daß *call*-*return*-Paare in α nur "wohlgeklammert" vorkommen: für jedes Präfix β von α und $q_0 \vdash_\beta \gamma$ ist $|\gamma| \geq 1$, da jede Ausführung eines *call* die Länge des Kellerinhalts γ' um 1 erhöht und jede Ausführung eines *return* sie um 1 erniedrigt. Es gibt daher wenigstens so viele *call*'s wie *return*'s in *jedem* β, und daher hat jedes *return* in α ein entsprechendes vorangegangenes *call*. Da man andererseits für $\beta = \alpha$ die Länge $|\gamma| = 1$ hat, gibt es umgekehrt für jedes *call* in α ein entsprechendes *return*.

Daher können wir den Beweis von $erase(\alpha) \in \mu X.\, L(\mathcal{A})[X/x]$ durch Induktion über die Schachtelungstiefe derjenigen *call*-*return*-Paare in α führen, deren *call* durch ein $(p, qq_0) \in \vdash'_{call} - \vdash_{call}$ interpretiert wird.

Falls es keine solchen Paare gibt, hat man $q_0 \vdash_\alpha q_e \in F$ in $\mathcal{A}$, und daher

$$w = erase(\alpha) \in L(\mathcal{A})[\emptyset/x] = L_1(\mathcal{A}) \subseteq \mu X.\, L(\mathcal{A})[X/x].$$

Sei nun die Schachtelungstiefe dieser *call*-*return*-Paare $n+1$. Sei $\alpha = \alpha_1 \cdot call \cdot \beta \cdot return \cdot \alpha_2$, wobei das angedeutete *call*-*return*-Paar durch ein $(p, qq'_0) \in \vdash'_{call} - \vdash_{call}$ interpretiert werde, es aber keine das $call \cdot \beta \cdot return$ umfassenden Paare dieser Art gebe. Dann ist

$$q_0 \vdash_{\alpha_1} \gamma p \vdash_{call} \gamma q q'_0 \vdash_\beta \gamma q q'_e \vdash_{return} \gamma q \vdash_{\alpha_2} q_e$$

für ein $q'_e \in F$. Durch Induktion folgt $erase(\beta) \in \mu X.\, L(\mathcal{A})[X/x]$ aus $q'_0 \vdash_\beta q'_e$. Nach Wahl von (p, qq'_0) ist $\gamma p \vdash_x \gamma q$ in $\mathcal{A}$. Da man dasselbe mit allen äußersten solchen Paaren machen kann, darf man annehmen,

$$q_0 \vdash_{\alpha_1} \gamma p \vdash_x \gamma q \vdash_{\alpha_2} q_e$$

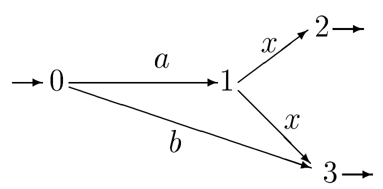
sei eine Ausführung mit $\mathcal{A}$. Also ist

$$\begin{array}{rcl} w = erase(\alpha) & = & erase(\alpha_1 x \alpha_2)[erase(\beta)/x] \\ & \in & L(\mathcal{A})[erase(\beta)/x] \\ & \subseteq & L(\mathcal{A})[\mu X.\, L(\mathcal{A})[X/x]/x] = \mu X.\, L(\mathcal{A})[X/x]. \end{array}$$

Der letzte Schritt gilt, da $\mu X.\, L(\mathcal{A})[X/x]$ ein Fixpunkt von $X \mapsto L(\mathcal{A})[X/x]$ ist. □

Wir können jetzt die Gleichung $\mu x(ax + b) = a^*b$ als Elimination der Endrekursion verstehen, wenn wir die Verwaltung von Rekursionsaufrufen betrachten:

Beispiel 7.4 Sei zunächst $\mathcal{A} = (\{0,1,2,3\}, \vdash_a, \vdash_b, \vdash_x, 0, \{2,3\})$ ein endlicher Automat über $\Sigma \cup \{x\}$, der $L(ax + b) = \{ax, b\}$ erkennt, mit den Übergangsrelationen wie in der folgenden Darstellung:



Der entsprechende rekursive endliche Automat $\mathcal{A}_{\mu x}$ hat statt $\vdash_x = \{(1,2),(1,3)\}$ die Relationen $\vdash_{call} = \{(1,30),(1,20)\}$ und $\vdash_{return} = \{(2,\varepsilon),(3,\varepsilon)\}$. Bei der Erkennung treten Rechnungen wie die folgende für das Wort *aaab* auf:

$$0 \vdash_a 1 \vdash_{call} 30 \vdash_a 31 \vdash_{call} 320 \vdash_a 321 \vdash_{call} 3220 \vdash_b 3223 \vdash_{return} 322 \vdash_{return} 32 \vdash_{return} 3.$$

Von den Rücksprungadressen wird kein wesentlicher Gebrauch gemacht, sie werden bloß zum Schluß vom Stapel entfernt. Man kann in solchen Fällen die Rekursionsaufrufe durch einfache Rücksprünge zum Anfangszustand ersetzen. Das ergibt hier einen endlichen Automaten $\mathcal{A}'$ über $\{a,b\}$ mit $\vdash_a = \{(0,0)\}$.

7.2 Kellerautomaten

Wir führen nun noch einen Automaten ein, der wie die rekursiven endlichen Automaten über einen Kellerspeicher verfügt. Allerdings wird der Keller nicht durch die Verwaltung rekursiver Aufrufe motiviert, sondern direkt durch die Erkennung von kontextfreien Sprachen durch Automaten. Durch eine einfache Ergänzung ergibt sich daraus ein Verfahren zur Syntaxanalyse mit kontextfreien Grammatiken.

Im speziellen Fall einer rechtslinearen Grammatik $G = (N, \Sigma, P, S)$, wo die Regeln ohne Einschränkung der Allgemeinheit von der Form $A \to aB$, $A \to a$ oder $A \to \varepsilon$ sind, mit $A, B \in N$ und $a \in \Sigma$, kann eine G-Ableitung als eine Berechnung eines endlichen Automaten verstanden werden. Dabei sind die Zustände gerade die syntaktischen Kategorien, und eine Verwendung der Regel $A \to aB$ entspricht einem Maschinenübergang $A \vdash_a B$. Um eine Eingabe $w = av$ als Ausdruck der Kategorie A zu erkennen, genügt es, das Restwort v als Ausdruck der Kategorie B zu erkennen. Wie weit die Analyse gediehen ist, wird durch die Konfiguartion $(B, v) \in N \times \Sigma^*$ beschrieben, wobei B die für die Resteingabe v erwartete Kategorie ist.

Bei einer kontextfreien Grammatik haben wir kompliziertere Regeln, etwa $A \to aBCD$ mit $a \in \Sigma$ und $B, C, D \in N$. Um hier eine Eingabe $w = av$ als Ausdruck der Kategorie A zu erkennen, genügt es entsprechend, das Restwort v als Folge von Wörtern v_1, v_2 und v_3 der Kategorien B, C und D zu erkennen. *Zuerst* muß also ein Präfix v_1 von v als Ausdruck der Kategorie B erkannt werden, *dann* ein Präfix v_2 der Resteingabe als Ausdruck der Kategorie C, und *schließlich* der verbliebene Rest v_3 als Ausdruck der Kategorie D. Wie weit die Analyse fortgeschritten ist, wird hier durch eine Konfiguration $(\gamma, v) \in N^* \times \Sigma^*$ beschrieben, wobei γ eine Folge von Kategorien für aufeinanderfolgende Teilworte der Resteingabe v ist. Haben wir v_1, v_2 und v_3 gefunden, so ist ein Anfangsstück $av_1v_2v_3$ von $w = av$ der Kategorie A gefunden. Um ein Präfix v_1 von v der Kategorie B zu finden, geht man auf dieselbe Weise mit einer Regel $B \to \beta \in P$ vor: in der Konfiguration (BCD, v) ersetzt man B durch β und arbeitet mit $(\beta CD, v)$ weiter. Beginnen β und v mit demselben Terminalsymbol $b \in \Sigma$, so verkürzt man die Folge βCD und die Resteingabe v um b.

Das legt nahe, die gesuchte Maschine mit folgenden Mitteln auszustatten:

(i) einen Stapel oder Kellerspeicher, in dem man sich eine beliebig lange Folge von syntaktischen Kategorien merken kann,

(ii) eine Operation, die die oberste Kategorie A des Stapels durch die rechte Regelseite einer Grammatikregel $A \to \alpha$ ersetzt;

(iii) eine Operation, die die Eingabe um den ersten Buchstaben und den Speicher um das oberste Symbol kürzt, falls diese übereinstimmen;

(iv) eine Entscheidung, ob $w \in L$, falls die Eingabe w völlig gelesen wurde.

Der Speicher und die Operationen darauf sind natürlich mit dem bei rekursiven endlichen Automaten verwendeten Stapel von Maschinenzuständen oder Rücksprungadressen und den Befehlen *call* und *return* verwandt. Wir arbeiten jetzt aber mit noch einfacheren Grundoperationen.

Definition 7.5 Ein STAPEL oder KELLERSPEICHER $(\Gamma^*, \{push_A, pop_A \mid A \in \Gamma\})$ besteht aus einem endlichen Alphabet Γ von Kellersymbolen und partiellen Operationen $push_A : \Gamma^* \to \Gamma^*$ und $pop_A : \Gamma^* \to \Gamma^*$, zu jedem $A \in \Gamma$, die einen KELLERINHALT $\gamma \in \Gamma^*$ wie folgt ändern:

$$push_A(\gamma) := A\gamma, \qquad pop_A(A\gamma) := \gamma, \qquad \text{undefiniert sonst.}$$

Ein (EIN-WEG) KELLERAUTOMAT $\mathcal{M} = (\Gamma, \Sigma, M, S)$ besteht aus endlichen Keller- und Eingabealphabeten Γ bzw. Σ, einem Anfangskellerinhalt $S \in \Gamma^*$, und einer regulären(!) Menge $M \subseteq \Sigma_\Gamma^*$ von Operationsfolgen über

$$\Sigma_\Gamma := \{push_A, pop_A, read_a \mid A \in \Gamma, a \in \Sigma\}.$$

Auf der Menge $\Gamma^* \times \Sigma^*$ der KONFIGURATIONEN VON $\mathcal{M}$ ist durch

$$\vdash_{\overline{M}} := \bigcup_{m \in M} \vdash_{\overline{m}}, \qquad \text{mit } \vdash_{\overline{m_1 \cdot m_2}} := \vdash_{\overline{m_1}} \circ \vdash_{\overline{m_2}} \text{ für } m_1, m_2 \in \Sigma_\Gamma^*,$$

eine Übergangsrelation gegeben, aufbauend auf den Übergangsrelationen

$$\begin{array}{lcl} \vdash_{\overline{pop_A}} & := & \{((A\gamma, w), (\gamma, w)) \mid \gamma \in \Gamma^*, w \in \Sigma^*\} \\ \vdash_{\overline{push_A}} & := & \{((\gamma, w), (A\gamma, w)) \mid \gamma \in \Gamma^*, w \in \Sigma^*\} \\ \vdash_{\overline{read_a}} & := & \{((\gamma, aw), (\gamma, w)) \mid \gamma \in \Gamma^*, w \in \Sigma^*\} \end{array}$$

für die Pop-, Push- und Lese-Operationen, mit $A \in \Gamma$ und $a \in \Sigma$. DIE VON $\mathcal{M}$ MIT LEEREM KELLER ERKANNTE SPRACHE ist

$$L(\mathcal{M}) := \{w \in \Sigma^* \mid (S, w) \vdash_{\overline{M}} (\varepsilon, \varepsilon)\}.$$

Den regulären Mengenoperationen entsprechen dann wie in Satz 2.5 die Vereinigung, das Relationenprodukt und die transitive, reflexive Hülle von Übergangsrelationen, d.h. es gilt

$$\begin{array}{lcllcll} \vdash_{\overline{M_1 \cup M_2}} & = & \vdash_{\overline{M_1}} \cup \vdash_{\overline{M_2}}, & \qquad \vdash_{\overline{0}} & = & \emptyset, & \\ \vdash_{\overline{M_1 \cdot M_2}} & = & \vdash_{\overline{M_1}} \circ \vdash_{\overline{M_2}}, & \qquad \vdash_{\overline{1}} & = & 1_{\Gamma^* \times \Sigma^*}, & \\ \vdash_{\overline{M^*}} & = & \vdash_{\overline{M}}{}^*, & \qquad \vdash_{\overline{\{m\}}} & = & \vdash_{\overline{m}} & \text{für } m \in \Sigma_\Gamma. \end{array}$$

Bemerkung 7.6 (i) Definition 7.5 ist nicht die Standarddefinition der Lehrbücher, sondern angelehnt an eine Arbeit von J. Goldstine[7]. Die folgenden Sätze zeigen, daß sie aber für die Spracherkennung zur üblichen Definition – vgl. z.B. Hopcroft/-Ullman[11] – äquivalent ist.

Die obige Definition legt nicht fest, wie die reguläre Menge M von Folgen aus Keller- und Leseoperationen dargestellt wird. Wir können M durch einen endlichen Automaten $\mathcal{A}_M$ mit $M = L(\mathcal{A}_M)$ angeben; dieser Automat ist die „endliche Kontrolle“, die die Zulässigkeit von Befehlsfolgen über Σ_Γ kontrolliert. Wir können M aber auch durch einen regulären Ausdruck r mit $M = L(r)$ angeben; dieser Ausdruck ist ein iteratives Programm über den Grundbefehlen $read_a$, $push_A$ und pop_A.

(ii) Die angegebene Definition von Kellermaschinen nutzt alle drei Interpretationen von regulären Ausdrücken aus Kapitel 2 aus: Die Maschinen sind modular aufgebaute Programme M, die einerseits durch $\vdash_{\overline{M}}$ als Relation auf der Gesamtheit aller Speicher × Eingabe– Konfigurationen interpretiert werden (= denotationelle Semantik) und andererseits operational als die Gesamtheit (= Sprache) aller Programmabläufe $m \in M$ — als Folgen von Basisoperationen — aufgefaßt werden.

Für die Erkennung von $L(G)$ kommt es nur auf $\vdash_{\overline{M}}$ an, aber für die Syntaxanalyse sind die verschiedenen Berechnungsfolgen $m \in M$ von Bedeutung.

Beachte, daß wir oben — im Unterschied zur Darstellung der endlichen Automaten — das Lesen eines Buchstabens $a \in \Sigma$ nicht durch eine Speicherveränderung $\vdash_a \subseteq \Gamma^* \times \Gamma^*$ beschrieben haben, sondern als eine (endlich kodierbare) Konfigurationsveränderung $\vdash_{\overline{read_a}} \subseteq (\Gamma^* \times \Sigma^*) \times (\Gamma^* \times \Sigma^*)$.

Beispiel 7.7 Die kontextfreie Sprache $L = \{a^n b^n \mid n \geq 0\}$ wird vom Kellerautomaten $\mathcal{M} = (\Sigma, \Sigma, M, \varepsilon)$ mit $M = (read_a; push_a)^*; (read_b; pop_a)^*$ erkannt: $L \subseteq L(\mathcal{M})$ gilt wegen

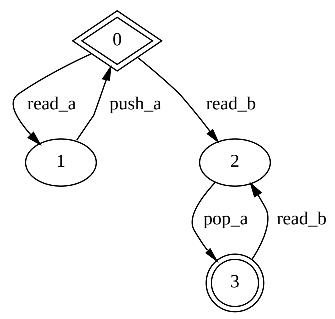
$$(\varepsilon, a^n b^n) \vdash_{\overline{(read_a;\, push_a)^*}} (a^n, b^n) \vdash_{\overline{(read_b;\, pop_a)^*}} (\varepsilon, \varepsilon).$$

Für $L(\mathcal{M}) \subseteq L$ sei $(\varepsilon, w) \vdash_{\overline{M}} (\varepsilon, \varepsilon)$. Dann gibt es $n, m \geq 0$ sowie $v \in \Gamma^*, u \in \Sigma^*$ mit

$$(\varepsilon, w) \vdash_{\overline{(read_a;\, push_a)^n}} (v, u) \vdash_{\overline{(read_b;\, pop_a)^m}} (\varepsilon, \varepsilon).$$

Daher ist $v = a^n \varepsilon$ und $w = a^n u$, sowie $u = b^m$ und $v = a^m$, also $w = a^n u = a^n b^m$ und $n = m$ wegen $a^n = v = a^m$. Also ist $w = a^n b^n \in L$.

Ein endlicher Automat $\mathcal{A}_M$, mit dem man die zulässigen Befehlsfolgen des Kellerautomaten erkennt, ist:



Mit der Konvention, daß der Kellerautomat mit leerem Keller startet und nur mit leerem Keller erkennt, kann man diese endliche Kontrolle $\mathcal{A}_M$ als eine vollständige Darstellung des Kellerautomaten ansehen.

7.3 Erkennung kontextfreier Sprachen mit Automaten

In den Abschnitten 9.2 und 9.3 konstruieren wir aus einer kontextfreien Grammatik G spezielle Kellerautomaten $\mathcal{M}^G$ und $\mathcal{M}_G$, die zur Syntaxanalyse bezüglich G geeignet sind. Vorher wird jetzt u.a. gezeigt, daß man mit beliebigen Kellerautomaten nicht mehr als kontextfreie Sprachen erkennen kann. Wie bei regulären Sprachen haben wir auch bei kontextfreien Sprachen verschiedene Definitions- und Erkennungsmöglichkeiten:

Satz 7.8 Sei $L \subseteq \Sigma^*$ eine Sprache über dem endlichen Alphabet Σ. Dann sind folgende Aussagen äquivalent:

(i) L ist durch eine reguläres Gleichungssystem S über Σ definierbar.

(ii) L ist durch einen μ-regulären Ausdruck r über Σ definierbar.

(iii) L wird von einem rekursiven endlichen Automaten $\mathcal{A} = (Q, \vdash_a, \vdash_{call}, \vdash_{return}, I, F)_{a \in \Sigma}$ erkannt.

(iv) L wird von einem Kellerautomaten $\mathcal{M} = (\Gamma, \Sigma, M, S)$ mit $S \in \Gamma$ erkannt.

(v) L wird von einer kontextfreien Grammatik $G = (N, \Sigma, P, S)$ erzeugt.

Beweis:

i) $\Rightarrow$ ii) $\Rightarrow$ iii) wurde in 6.13 und 7.3 gezeigt.

iii) $\Rightarrow$ iv): Wir geben dazu M durch einen regulären Ausdruck über Σ_Γ an, also durch ein iteratives Programm r mit den Grundbefehlen für Kellerautomaten. Dann ist $M = L(r) \subseteq \Sigma_\Gamma^*$ regulär.

Wähle $\Gamma := Q \,\dot{\cup}\, \{S\}$ und $r := \sum_{p \in I}(pop_S \cdot push_p) \cdot s^* \cdot (\sum_{q \in F} pop_q)$ mit

$$s \quad := \quad \sum_{p \,\vdash_a\, q} (read_a \cdot pop_p \cdot push_q) \; + \sum_{p \,\vdash_{return}\, \varepsilon} pop_p \; + \sum_{p \,\vdash_{call}\, rq} (pop_p \cdot push_r \cdot push_q)$$

Ist $w \in L(\mathcal{M})$, also $(S, w) \vdash_r^{\mathcal{M}} (\varepsilon, \varepsilon)$, so gibt es ein $p \in I, q \in F$ mit $(p, w) \vdash_{s^*}^{\mathcal{M}} (q, \varepsilon)$. (O.B.d.A. sei $I \cap F = \emptyset$, also $p \neq q$.) Nach Konstruktion simuliert s die Grundbefehle $\vdash_a$, $\vdash_{call}$ und $\vdash_{return}$ von $\mathcal{A}$. Daher gibt es $\alpha \in (\Sigma \cup \{call, return\})^*$ mit $(p, w) \vdash_\alpha^{\mathcal{A}} (q, \varepsilon)$ und $w = erase(\alpha)$, so daß $w \in L(\mathcal{A})$. Umgekehrt gibt es zu $w \in L(\mathcal{A})$ ein α mit $w = erase(\alpha)$

und $p \vdash_{\alpha}^{\mathcal{A}} q$ für geeignete Zustände $p \in I, q \in F$. Da man α durch eine Befehlsfolge aus $L(s^*)$ simulieren kann, erhält man

$$(S, w) \vdash_{pop_S}^{\mathcal{M}} (\varepsilon, w) \vdash_{push_p}^{\mathcal{M}} (p, w) \vdash_{s^*}^{\mathcal{M}} (q, \varepsilon) \vdash_{pop_q}^{\mathcal{M}} (\varepsilon, \varepsilon),$$

so daß w durch eine zulässige Befehlsfolge von $\mathcal{M}$ akzeptiert wird.

iv) $\Rightarrow$ v): Da $M \subseteq \Sigma_\Gamma^*$ regulär ist, gibt es einen deterministischen endlichen Automaten $\mathcal{A}_M = (Q_M, \Sigma_\Gamma, \delta, q_0, F)$, der M erkennt. In einer syntaktischen Kategorien einer Grammatik für L fassen wir die Eingabewörter w zusammen, die durch eine $\mathcal{M}$-Berechnung $m \in M$ erkannt werden, die zu deren Erkennung mit $\mathcal{M}$ nur ein Kellersymbol X verbraucht wird

Definiere eine kontextfreie Grammatik $G = (N, \Sigma, P, S_G)$ mit der Nonterminalmenge

$$N := \{S_G\} \cup \{\, [p, X, q] \mid p, q \in Q_M, X \in \Gamma \cup \{\varepsilon\} \,\}$$

und der folgenden Menge P von Produktionsregeln, für alle $r, s \in Q_M$, $A \in \Gamma$ und $X \in \Gamma \cup \{\varepsilon\}$:

$$\begin{array}{lcll}
S_G & \to & [q_0, S, q], & \text{für } q \in F, \\
[r, \varepsilon, r] & \to & \varepsilon, & \\
[p, X, r] & \to & a[q, X, r], & \text{für } \delta(p, read_a) = q, \\
[p, A, r] & \to & [q, \varepsilon, r], & \text{für } \delta(p, pop_A) = q, \\
[p, X, r] & \to & [q, A, s][s, X, r], & \text{für } \delta(p, push_A) = q.
\end{array}$$

Beh.: Für jedes $w \in \Sigma^*$ und $X \in \Gamma \cup \{\varepsilon\}$ sind folgende Aussagen äquivalent:

a) $[p, X, r] \overset{*}{\Longrightarrow}_G w$

b) Es gibt $m \in \Sigma_\Gamma^*$ mit $\delta(p, m) = r$ und $(X, w) \vdash_{\overline{m}} (\varepsilon, \varepsilon)$.

Hieraus folgt die Aussage des Satzes über

$$\begin{array}{rcl}
w \in L(G) & \iff & S_G \overset{*}{\Longrightarrow}_G w \\
& \iff & \exists q \in F \; [q_0, S, q] \overset{*}{\Longrightarrow} w, \\
& \iff & \exists m \in M \; (S, w) \vdash_{\overline{m}} (\varepsilon, \varepsilon), \\
& \iff & w \in L(\mathcal{M}).
\end{array}$$

a) $\Longrightarrow$ b): Durch Induktion über die Ableitungslänge n von $[p, X, r] \overset{n}{\Longrightarrow}_G w$. (Übung!)

b) $\Longrightarrow$ a): Durch Induktion über die Länge $|m|$ der Operationsfolge m:

$m \equiv \varepsilon \in \Sigma_\Gamma^*$: Dann ist $p = r$ und $(X, w) = (\varepsilon, \varepsilon)$. Also ist $[p, X, r] \to w$ die Regel $[r, \varepsilon, r] \to \varepsilon$ aus P, und daher $[p, X, r] \overset{1}{\Longrightarrow}_G w$.

$m \equiv read_a \cdot m_0$: Dann ist $(X,w) \vdash_{\overline{read_a}} (X,v) \vdash_{\overline{m_0}} (\varepsilon,\varepsilon)$, also $w \equiv av$, und zu $q = \delta(p, read_a)$ gibt es eine G-Regel $[p,X,r] \longrightarrow a[q,X,r]$. Aus der Induktionsannahme folgt dann

$$[p,X,r] \overset{1}{\underset{G}{\Longrightarrow}} a[q,X,r] \overset{*}{\underset{G}{\Longrightarrow}} av \;\equiv w.$$

$m \equiv pop_A \cdot m_0$: Sei $q = \delta(p, pop_A)$. Nach der Voraussetzung ist $(X,w) \vdash_{\overline{pop_A \cdot m_0}} (\varepsilon,\varepsilon)$, weshalb $X \equiv A$ und $(\varepsilon, w) \vdash_{\overline{m_0}} (\varepsilon,\varepsilon)$ gelten muß. Wegen $\delta(q,m_0) = r$ ist induktiv schon $[q,\varepsilon,r] \overset{*}{\underset{G}{\Longrightarrow}} w$, also auch

$$[p,X,r] \overset{1}{\underset{G}{\Longrightarrow}} [q,\varepsilon,r] \overset{*}{\underset{G}{\Longrightarrow}} w.$$

$m \equiv push_A \cdot m_0$: Sei $q = \delta(p, push_A)$, also $r = \delta(q, m_0)$. Wegen $(X,w) \vdash_{\overline{push_A \cdot m_0}} (\varepsilon,\varepsilon)$ ist $(AX,w) \vdash_{\overline{m_0}} (\varepsilon,\varepsilon)$. Da diese $\mathcal{M}$-Berechnung den Keller leert, muß es Zerlegungen $m_0 = m_1m_2$ und $w \equiv w_1w_2$ geben, so daß

$$(AX, w_1w_2) \vdash_{\overline{m_1}} (X, w_2) \vdash_{\overline{m_2}} (\varepsilon,\varepsilon)$$

und die Ausführung von m_1 den Teil X des Kellerinhalts nicht entfernt. Dann ist m_1 auch eine Berechnung für $(A, w_1) \vdash_{\overline{m_1}} (\varepsilon,\varepsilon)$. Nach Induktionsannahme gilt daher mit $s = \delta(q, m_1)$ schon $[q,A,s] \overset{*}{\underset{G}{\Longrightarrow}} w_1$ und $[s,X,r] \overset{*}{\underset{G}{\Longrightarrow}} w_2$. Zu $push_A$ gibt es eine G-Regel

$$[p,X,r] \longrightarrow [q,A,s][s,X,r],$$

mit der wir insgesamt eine Ableitung

$$[p,X,r] \overset{1}{\underset{G}{\Longrightarrow}} [q,A,s][s,X,r] \overset{*}{\underset{G}{\Longrightarrow}} w_1[s,X,r] \overset{*}{\underset{G}{\Longrightarrow}} w_1w_2 \;\equiv w$$

der gewünschten Art bilden können.

v) $\Rightarrow$ i): Sei $P = \{\, x_i = r_i \mid 1 \le i \le m \,\}$ das der Regelmenge P von G entsprechende Gleichungssystem und $(A_1, \ldots, A_m)$ seine kleinste Lösung in $\mathcal{L}_\Sigma$. Durch Induktion über n zeigt man für alle $w \in \Sigma^*$und $1 \le i \le m$:

$$x_i \overset{\le n}{\underset{G}{\Longrightarrow}} w \quad \Longrightarrow \quad w \in A_i.$$

Sei $(A_{1,n}, \ldots, A_{m,n}) = F_P^n(\emptyset, \ldots, \emptyset)$ das n-te Stadium der Berechnung von $(A_1, \ldots, A_m)$. Für $n = 1$ ist w ein Summand von r_i und daher $w \in A_{i,1}$. Für $n > 0$ sei

$$x_i \underset{G}{\Longrightarrow} Y_1 \cdots Y_k \overset{\le n}{\underset{G}{\Longrightarrow}} w.$$

Dann gibt es eine Zerlegung $w = w_1 \cdots w_k$, so daß $Y_j \overset{\le n}{\underset{G}{\Longrightarrow}} w_j$ für die Variablen Y_j unter $(Y_1, \ldots, Y_k)$ und $Y_j \overset{0}{\underset{G}{\Longrightarrow}} w_j$ für die Konstanten. Nach Induktion ist $w_j \in A_{j,n}$, also

$$w = w_1 \cdots w_k \in r_i^{\mathcal{L}_\Sigma}(A_{1,n}, \ldots, A_{m,n}) = A_{i,n+1} \subseteq A_i,$$

da $Y_1 \cdots Y_k$ ein Summand von r_i ist. □

Die im Beweis angegebene kontextfreie Grammatik hat nur Regeln einer einfachen, aber nicht mehr linearen Form:

Korollar 7.9 Jede gleichungsdefinierbare Sprache kann durch ein Gleichungssystem

$$\begin{array}{ccc} x_1 & = & r_1(x_1, \ldots, x_m) \\ \vdots & \vdots & \vdots \\ x_m & = & r_m(x_1, \ldots, x_m) \end{array}$$

definiert werden, worin jedes r_i eine Summe von Ausdrücken der Form $1, x_j, ax_j$ und x_jx_k ist, mit $a \in \Sigma$ und $1 \leq j, k \leq m$.

Beachte, daß alle Summanden höchstens die Länge 2 haben. Allerdings haben wir Summanden mit führenden Variablen, d.h. solchen der Form $x_i\alpha$; sonst wäre das System rechtslinear. In diesem Fall kann man nach Lemma **??** einen erkennenden endlichen Automaten konstruieren, indem man die Summanden des Gleichungssystems auf eine einfache Form bringt; dabei werden insbesondere die Summanden der Form x_i durch solche der Form ax_j mit $a \in \Sigma$ ersetzt. Das besagte, daß die erkennenden Automaten keine spontanen, von der Eingabe unabhängigen Schritte ausführen.

Für die Erkennung kontextfreier Sprachen ist die Beseitigung von Summanden mit führender Variablen von entsprechender Bedeutung. Das leistet die Greibach-Normalform aus Korollar 6.36: dort werden aus der Gleichungsdefinition die Summanden mit führenden Variablen durch Summanden der Form $ax_j\beta$ mit $a \in \Sigma$ ersetzt. Nach jedem rekursiven Aufruf (des der Grammatik entsprechenden rekursiven endlichen Automaten) muß daher ein Lesen in der Eingabe erfolgen.

Aufgaben

Aufgabe 7.1 Man gebe Kellerautomaten $\mathcal{M}_i = (\Sigma, \Gamma, M_i)$ an, die folgende Sprachen erkennen:

(i) $L_2 = \{w \in \{a, b\}^* \mid |w|_a = 2 \cdot |w|_b\}$, wobei $|w|_a$ die Anzahl der Vorkommen von a im Wort w ist.

(ii) $L_1 = \{a^n b^m \mid 0 \leq n \leq m \leq 2n\}$.

(iii) Die reguläre Menge M_1 oder $M_2 \subseteq \Sigma_\Gamma$ gebe man sowohl durch einen regulären Ausdruck als auch durch einen endlichen Automaten (über dem Alphabet Σ_Γ) an.

Aufgabe 7.2 Man zeige, daß die Kellerautomaten nach 7.5 dieselben Sprachen erkennen wie die im Lehrbuch von Hopcroft und Ullman[11] definierten Kellerautomaten.

Aufgabe 7.3 (i) Man gebe zur Teilgrammatik aus Aufgabe 6.2 einen rekursiven endlichen Automaten $\mathcal{A}$ (mit Bild) an, der die Sprache $L(\mathrm{NP})$ erkennt.

(ii) Zeige an einer akzeptierenden Berechnungsfolge von $\mathcal{A}$ für den Beispielausdruck (*) aus Aufgabe 6.2, wie sich der Stapel der (Rücksprung-) Zustände von $\mathcal{A}$ verhält.

Aufgabe 7.4 Sei $\Sigma = \{a, b, c\}$ und S das Gleichungssystem

$$\begin{array}{rcl} x_1 & = & ax_1 + bx_2 + c \quad =: \quad r_1(x_1, x_2) \\ x_2 & = & bx_1 + bx_2 + c \quad =: \quad r_2(x_1, x_2) \end{array}$$

über Σ. Sei L_i die x_i-Komponente der kleinsten Lösung bei der Sprachinterpretation $\mathcal{L}$.

(i) Man gebe μ-reguläre Ausdrücke s_i mit $L(s_i) = L_i$ an.

(ii) Benutze $\mathcal{L} \models \forall y \forall z\, (\mu x.(yx + z) = y^* z)$, um aus dem μ-regulären Ausdruck s_1 aus (i) einen äquivalenten regulären Ausdruck zu machen.

(iii) Man gebe endliche Automaten

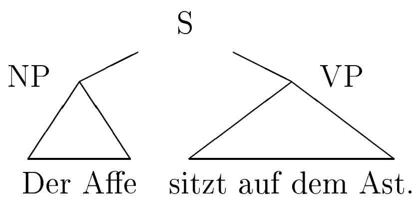
$$\mathcal{A}_i = (Q_i, \Sigma \cup \{x_1, x_2\}, \vdash_a, \vdash_b, \vdash_c, q_0^i, F^i)$$

und ihre Darstellung als Graphen an, die die Sprachen $L(s_i) \subseteq (\Sigma \cup \{x_1, x_2\})^*$ erkennen.

(iv) Konstruiere aus $\mathcal{A}_1, \mathcal{A}_2$ aus Teil (iii) einen rekursiven endlichen Automaten $\mathcal{A}$, der L_1 erkennt.

8 Die Phrasenstruktur kontextfreier Sprachen

Die Grundauffassung bei einer Sprachbeschreibung durch kontextfreie Grammatiken besteht darin, daß man sich die Sätze $a_1 \ldots a_n$ aus *zusammenhängenden* Konstituenten $a_i a_{i+1} \ldots a_{j-1} a_j$ gebildet denkt, und Konstituenten *hierarchisch* aus Teilkonstituenten erzeugt. Daher erhält man als syntaktische Struktur stets *angeordnete Bäume*:



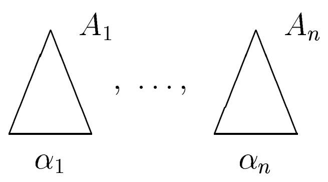
Jeder Ableitung $S \overset{*}{\Longrightarrow} \alpha$ einer kontextfreien Grammatik kann ein Baum zugeordnet werden; ein Baum repräsentiert dann anschaulich die Äquivalenzklasse aller Ableitungen, denen dieser Baum zugeordnet wurde.

8.1 Ableitungsbäume kontextfreier Grammatiken

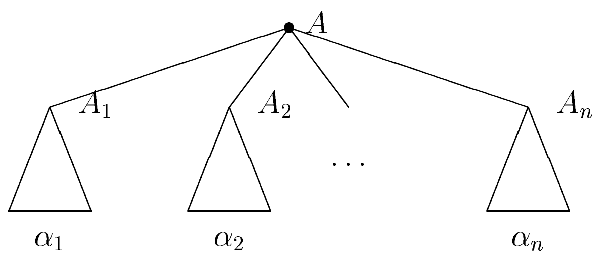
Anschaulich wollen wir die Menge der Ableitungsbäume zur kontextfreien Grammatik G induktiv durch die folgenden Bedingungen definieren:

(i) Ist $\bullet$ ein Punkt und $A \in N$, so ist der einpunktige Baum $\bullet^A$ ein G-Ableitungsbaum.

(ii) Ist $A \longrightarrow A_1 \cdots A_n$ eine Regel von G und sind



schon G-Ableitungsbäume, so ist auch



ein G-Ableitungsbaum.

In den folgenden beiden Definitionen wird diese Vorstellung genauer formuliert. Zuerst werden (endliche) Bäume mit einer Markierung der Knoten und einer Anordnung der Äste eingeführt. Danach legen wir fest, was wir mit dem Ableitungsbaum zu einer gegebenen G-Ableitung meinen.

Definition 8.1 Sei $t = (T, \leq, 0)$ ein Baum. Eine MARKIERUNG VON t über M ist eine Abbildung $m : T \longrightarrow M$, die jedem Punkt $x \in T$ eine Marke $m(x) \in M$ zuordnet. Eine ANORDNUNG von t ist eine 2-stellige Relation $\prec$ auf T, so daß für jedes $x \in T$ die Menge der direkten $\leq$-Nachfolger von x durch $\prec$ total geordnet wird. Ein GEORDNETER, MARKIERTER BAUM $(t, \prec, m)$ ist ein Baum t mit einer Anordnung $^\prec$ und einer Markierung m.

Beispiel 8.2 Als ungeordnete Bäume sind

```
        0                         0
       / \                       / \
      1   2         und         2   1
     / \  |                     |  / \
    3   4 5                     5 3   4
```

gleich. Ordnet man die $\leq$-Nachfolger entsprechend der links-rechts-Reihenfolge in der Darstellung, so erhält man die verschieden geordneten Bäume

$$(t_1, \prec_1) = (t_1, \{(1,2), (3,4)\}) \quad \text{und} \quad (t_2, \prec_2) = (t, \{(2,1), (3,4)\}).$$

Definition 8.3 Sei $G = (N, \Sigma, P, S)$ eine kontextfreie Grammatik. Ein G-ABLEITUNGSBAUM ist ein geordneter, markierter Baum $\tau = ((T, \leq, 0), \prec, m)$ mit den Eigenschaften:

(i) $m : T \longrightarrow N \cup \Sigma \cup \{\varepsilon\}$, und

(ii) Sind $x_1 \prec \ldots \prec x_n$ die direkten $\leq$-Nachfolger eines Punktes $x \in T$, so gibt es eine Regel $A \rightarrow A_1 \ldots A_n \in P$ mit $m(x) = A$ und $m(x_i) = A_i$ für $i = 1, \ldots, n$, wobei $A_i = \varepsilon$ nur für den Fall $n = 1$ erlaubt ist.

Jeder G-Ableitung $A \Longrightarrow_G^n \alpha$ ordnen wir induktiv über n einen (bzw. eine Klasse isomorpher) markierten, geordneten Ableitungsbaum zu:

(i) Ein ABLEITUNGSBAUM FÜR $A \Longrightarrow_G^0 A$ ist ein geordneter, markierter Baum $\tau = ((T, \leq, 0), \prec, m)$ mit einem einpunktigen Baum $(T, \leq, 0) = (\{0\}, (0,0), 0)$ und der Markierung $m(0) = A \in N \cup \Sigma$. Da es keine direkten $\leq$-Nachfolger der Wurzel $0 \in T$ gibt, ist die Anordnung $\prec$ leer.

(ii) Sei $A \overset{n}{\underset{G}{\Longrightarrow}} \alpha B\gamma \overset{1}{\underset{G}{\Longrightarrow}} \alpha\beta\gamma$ und $\tau_A = ((T_A, \leq_A, 0_A), \prec_A, m_A)$ ein G-Ableitungsbaum für eine Ableitung zu $A \overset{n}{\underset{G}{\Longrightarrow}} \alpha B\gamma$, $\tau_B = ((T_B, \leq_B, 0_B), \prec_B, m_B)$ ein G-Ableitungsbaum für die Ableitung $B \overset{1}{\underset{G}{\Longrightarrow}} \beta$. Wir können annehmen, daß $T_A \cap T_B = \{0_B\}$ und daß 0_B das Blatt des Baums T_A ist, das mit dem ausgezeichneten B in $\alpha B\gamma$ markiert ist. Dann ist

$$\tau = ((T_A \cup T_B,\ (\leq_A \cup \leq_B)^*,\ 0_A),\ \prec_A \cup \prec_B,\ m_A \cup m_B)$$

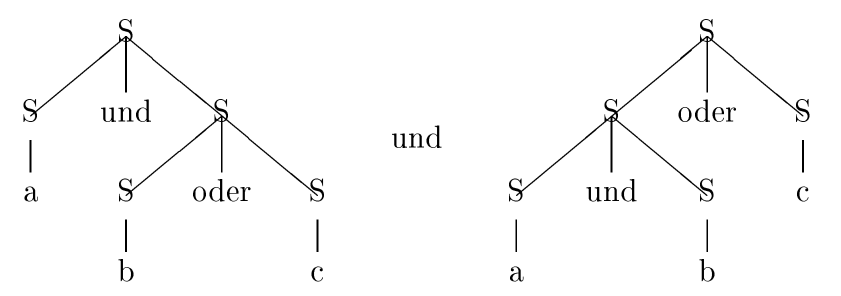
ein G-Ableitungsbaum für $A \overset{n+1}{\underset{G}{\Longrightarrow}} \alpha\beta\gamma$, bzw. für die aus den beiden gegebenen Ableitungen zusammengesetzte Ableitung zu $A \overset{n}{\underset{G}{\Longrightarrow}} \alpha B\gamma \overset{1}{\underset{G}{\Longrightarrow}} \alpha\beta\gamma$.

Zwei G-Ableitungsbäume $\tau_i = ((T^{\tau_i}, \leq^{\tau_i}, 0^{\tau_i}),\ \prec^{\tau_i}, m^{\tau_i})$, $i = 1, 2$, heißen isomorph, wenn es eine bijektive Abbildung $f : T^{\tau_1} \to T^{\tau_2}$ gibt, die die Wurzeleigenschaft, die partielle Ordnung $\leq$, die Markierung und die Anordnung $\prec$ erhält, d.h. wenn für alle $x, y \in T^{\tau_1}$ gilt:

$$\begin{array}{rcl@{\qquad}rcl} f(0^{\tau_1}) & = & 0^{\tau_2}, & x \leq^{\tau_1} y & \Longleftrightarrow & f(x) \leq^{\tau_2} f(y), \\ m^{\tau_1}(x) & = & m^{\tau_2}(f(x)), & x \prec^{\tau_1} y & \Longleftrightarrow & f(x) \prec^{\tau_2} f(y). \end{array}$$

Ein Wort $w \in L(G)$ heißt syntaktisch mehrdeutig oder ambig, wenn es zwei nicht-isomorphe Ableitungsbäume mit Wurzel S und Ergebnis w gibt. Eine Grammatik G ist ambig, wenn mindestens ein $w \in L(G)$ ambig ist.

Beispiel 8.4 Sei $P = \{S \to a,\ S \to b,\ S \to c,\ S \to S \text{ und } S,\ S \to S \text{ oder } S\}$. Dann ist der Satz "a und b oder c" mehrdeutig, da er zwei Ableitungsbäume hat, nämlich



Zwei G-Ableitungen betrachten wir als äquivalent, wenn es isomorphe G-Ableitungsbäume für sie gibt. Ableitungsbäume repräsentieren also Klassen äquivalenter Ableitungen: sie abstrahieren von Unterschieden in der Reihenfolge der Regelanwendungen.

Eine Grammatik G heißt (un)beschränkt mehrdeutig, wenn es (k)eine Schranke $k \in \mathbb{N}$ gibt, so daß jedes $w \in L(G)$ höchstens k viele verschiedene G-Ableitungsbäume hat.

Beispiel 8.5 Den regulären Gleichungssystemen

$$\begin{array}{rcl} S & = & A+B \\ A & = & a+Ab \\ B & = & a+Bb \end{array} \qquad \text{bzw.} \qquad \begin{array}{rcl} S & = & A\cdot B \\ A & = & a+Ab \\ B & = & a+bB \end{array}$$

entsprechen eine beschränkt mehrdeutige Grammatik G_1 bzw. eine unbeschränkt mehrdeutige Grammatik G_2. Jedes $ab^n \in L(G_1)$ hat eine Analyse als Ausdruck der Kategorie A und eine als Ausdruck der Kategorie B. Jedes $ab^k a \in L(G_2)$ kann auf $k+1$ Arten in ein Präfix ab^n der Kategorie A und ein Suffix $b^m a$ der Kategorie B zerlegt werden.

8.2 Wiederholungseigenschaften kontextfreier Sprachen

Wir wollen nun Wiederholungen von Teilwörtern in Wörtern einer kontextfreien Sprache untersuchen. Dazu zerlegen wir Ableitungsbäume in Teilbäume:

Definition 8.6 Sei $G = (N, \Sigma, P, S)$ eine kontextfreie Grammatik und τ ein G-Ableitungsbaum.

(i) Die $\textsc{Front}(\tau)$ ist die Verkettung der Blättermarken an τ, in der durch $\prec$ gegebenen Ordnung der Zweige.

(ii) τ ist ein $\textsc{Basisbaum}$ mit Wurzel $A :\Longleftrightarrow$

- (a) Wurzelmarke$(\tau) = A$
- (b) Front$(\tau) \in \Sigma^*$
- (c) Jedes $B \in N$ kommt auf jedem Pfad in τ höchstens einmal vor.

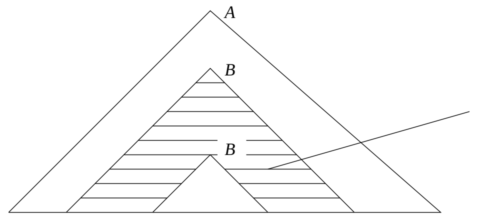
(iii) τ ist ein A-Baum: $:\Longleftrightarrow$

- (a) Wurzelmarke$(\tau) = A \in N$
- (b) Front$(\tau) \in \Sigma^* A \Sigma^*$
- (c) Jedes $B \in N, B \neq A$, kommt auf jedem Pfad von τ höchstens einmal vor
- (d) A kommt auf jedem Pfad von τ höchstens zweimal vor.

Lemma 8.7 Sei τ ein G-Ableitungsbaum mit Wurzelmarke A, und Front$(\tau) \in \Sigma^*$. Dann gilt

(i) τ ist Basisbaum mit Wurzel A, oder

(ii) Es gibt $B \in N$ und einen B-Baum τ_B, so daß τ die folgende Form hat:

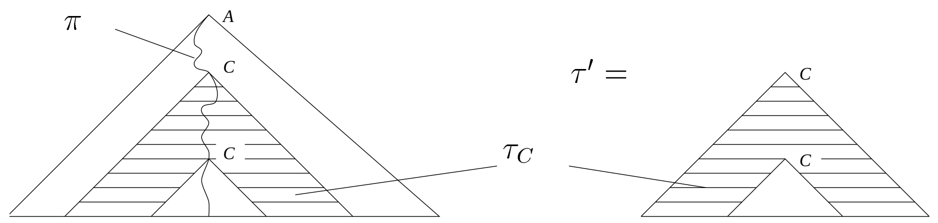
$\tau =$ 

τ_B.

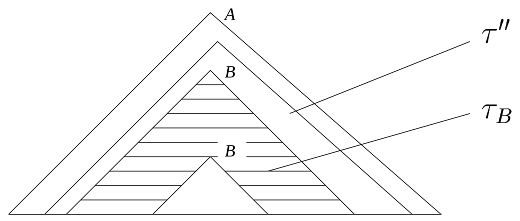
Beweis: Durch Induktion über die Größe $|\tau|$ von τ.

$|\tau| = 1$: Dann ist τ ein Basisbaum mit Wurzel A.

$|\tau| > 1$: Angenommen, τ sei kein Basisbaum mit Wurzel A. Dann gibt es einen Pfad π in τ, auf dem ein $C \in N$ mindestens zweimal vorkommt. Wähle τ' und τ_C wie in

$\tau =$ 

Falls τ_C ein C-Baum ist, gilt ii) schon. Andernfalls kommt ein $D \in N$ mit $D \neq C$ zweimal auf einem Pfad in τ_C vor, oder C kommt dreimal auf einem Pfad in τ_C vor. In beiden Fällen gibt es einen *echten* Teilbaum τ'' von τ', in dem ein Nonterminal D mindestens zweimal auf einem Pfad vorkommt. Also ist τ'' kein Basisbaum. Nach Induktionsannahme gibt es wegen $|\tau''| < |\tau|$ daher ein $B \in N$ und einen B-Baum τ_B, so daß

$\tau =$ 

Daher gilt ebenfalls ii). □

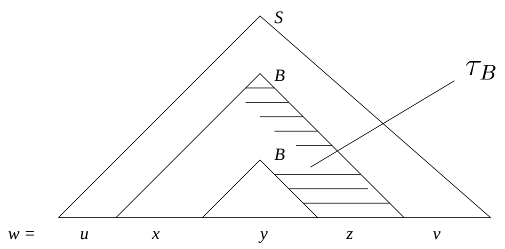
Lemma 8.8 (Wiederholungs- oder Pumping-Lemma für Typ 2-Sprachen) Zu jeder kontextfreien Sprache L gibt es Konstanten $p, q \in I\!N$, so daß für alle $w \in L$ mit $|w| > p$ eine Zerlegung $w = uxyzv$ existiert, so daß

$$i)\ xz \neq \varepsilon, \quad ii)\ |xyz| \leq q \quad \text{und} \quad iii)\ \{u\ x^n y\ z^n\ v \mid n \geq 0\} \subseteq L.$$

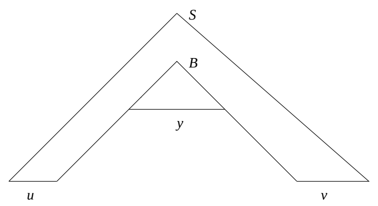
Beweis: Sei $G = (N, \Sigma, P, S)$ eine kontextfreie Grammatik für L, und

$$p := \max\{\, |Front(\tau)| \mid \tau \text{ ist } G\text{-Basisbaum mit Wurzel } S\,\}.$$

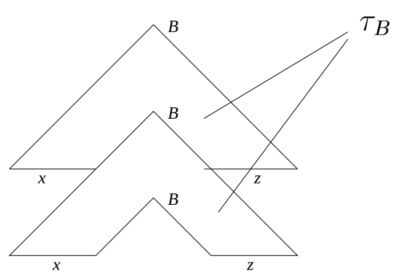
Sei τ ein G-Ableitungsbaum für w, und τ_B ein Teilstück davon wie im vorigen Lemma:

$\tau =$ 

Wir können annehmen, daß $xz \neq \varepsilon$, da wir sonst den kleineren Ableitungsbaum



betrachten. Außerdem können wir annehmen, daß der Teilbaum mit Wurzel B und Front y ein Basisbaum von G ist; andernfalls suchen wir einen entsprechenden Teilbaum davon. In τ können wir (wegen $B \overset{*}{\Longrightarrow}_G x^n\ B\ z^n$) den B-Baum τ_B durch Vielfache τ_B^n davon, mit

$\tau_B^0 := \cdot B, \quad \tau_B^1 := \tau_B, \quad \tau_B^2 :=$  usw.,

ersetzen und erhalten so ein Folge von G-Ableitungsbäumen für die Wörter ux^nyz^nv. Also ist $\{ux^nyz^nv \mid n \geq 0\} \subseteq L(G) = L$. Beachte, daß sich aus der Frontlänge von Basisbäumen und A-Bäumen von G eine obere Schranke q für $|xyz|$ ergibt. $\square$

Folgerung 8.9 Sei G eine kontextfreie Grammatik. $L(G)$ ist genau dann unendlich, wenn ein Basisbaum τ mit Wurzel S und eine Marke A in τ existiert, zu der es einen A-Baum τ_A von G mit $|Front(\tau_A)| > 1$ gibt.

Mit dem Wiederholungslemma können wir von gewissen Sprachen zeigen, daß sie *nicht* kontextfrei sind:

Beispiel 8.10 $L = \{a^n b^n c^n \mid n > 0\}$ ist nicht kontextfrei.

Beweis: Sei $a^n b^n c^n = uxyzv$ bei geeignet großem n, mit u, x, y, z, v wie im Wiederholungslemma. Falls x oder z verschiedene Buchstaben (d.h. a und b, b und c) enthält, ist $ux^2yz^2v \notin L$, also $L \notin \mathcal{CFL}$ nach dem Wiederholungslemma. Falls x und z nur aus einer Sorte Buchstaben besteht, ist ebenfalls $ux^2yz^2v \notin L$, da der dritte Buchstabe nicht verdoppelt wurde. □

8.3 Worthäufigkeit und -stellung in kontextfreien Sprachen

Für die Untersuchung von Sprachen mit freier Wortstellung ist lediglich die Anzahl der verschiedenen Wörter in einem Satz von Bedeutung, nicht ihre Reihenfolge. Für kontextfreie Sprachen hat R. Parikh untersucht, wie häufig die Wörter in den Sätzen (bzw. die Buchstaben in den Wörtern) der Sprache vorkommen.

Definition 8.11 Sei $\Sigma = \{a_1, \ldots, a_n\}$. Für $a \in \Sigma$, $w \in \Sigma^*$, und $L \subseteq \Sigma^*$ sei $|w|_a$ die Anzahl der Vorkommen von a in w,

$$\|w\|_\Sigma := (|w|_{a_1}, \ldots, |w|_{a_n})$$

die BUCHSTABENHÄUFIGKEIT VON w, und $\|L\|_\Sigma := \{\, \|w\|_\Sigma \mid w \in L \,\}$ die BUCHSTABENHÄUFIGKEIT VON L oder das KOMMUTATIVE BILD von L.

Eine Menge $M \subseteq \mathbb{N}^n$ heißt LINEAR, falls es Tupel $v_0, \ldots, v_m \in \mathbb{N}^n$ gibt, so daß

$$M = \{\, v_0 + \sum_{i=1}^{m} k_i v_i \mid k_1, \ldots, k_m \in \mathbb{N} \,\},$$

wobei $k \cdot (l_1, \ldots, l_n) := (k \cdot l_1, \ldots, k \cdot l_n)$ und $(k_1, \ldots, k_n) + (l_1, \ldots, l_n) := (k_1 + l_1, \ldots, k_n + l_n)$. M heißt SEMILINEAR, falls M eine endliche Vereinigung linearer Mengen ist.

Satz 8.12 (Parikh 1966)

(i) Ist $M \subseteq I\!N^n$ semilinear, so gibt es eine reguläre Sprache $L \subseteq \Sigma^*$ mit $M = \|L\|_\Sigma$.

(ii) Ist $L \subseteq \Sigma^*$ kontextfrei, so ist $\|L\|_\Sigma$ semilinear.

Beweis: Sei $\Sigma = \{a_1, \ldots, a_n\}$. Wir schreiben $\|L\|$ statt $\|L\|_\Sigma$.

(i) Wegen $\|L_1 \cup L_2\| = \|L_1\| \cup \|L_2\|$ genügt es, die Behauptung für lineares M zu zeigen. Sei $M = \{\, v_0 + \Sigma_{i=1}^m k_i \cdot v_i \mid k_i \in I\!N \,\}$ mit $v_j = (l_{j,1}, \ldots, l_{j,n})$. Wähle

$$r = a_1^{l_{0,1}} \ldots a_n^{l_{0,n}} \cdot (a_1^{l_{1,1}} \ldots a_n^{l_{1,n}})^* \ldots (a_1^{l_{n,1}} \ldots a_n^{l_{m,n}})^*$$

mit $a_j^k = a_j \ldots a_j$ (k-mal). Offenbar ist $\|L(r)\| = M$.

(ii) Sei $G = (N, \Sigma, P, S)$ eine kontextfreie Grammatik für L und $NT(\tau)$ die Menge der Nonterminale eines G-Ableitungsbaums τ. Sei $\mathcal{F}$ die Menge aller endlichen Folgen $F = (\tau_o^F, \ldots, \tau_{k^F}^F)$ von G-Ableitungsbäumen mit

a) τ_0^F ist Basisbaum mit Wurzel S,

b) τ_i^F ist A_i-Baum für ein $A_i \in \bigcup_{j<i} NT(\tau_j^F)$,

c) $\tau_i^F \notin \{\tau_0^F, \ldots, \tau_{i-1}^F\}$.

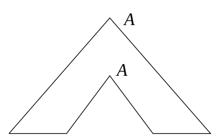
Beachte, daß $\mathcal{F}$ endlich ist: die Anzahl der Basis- und A-Bäume ist beschränkt, da ihre Tiefe höchstens $|N| + 2$ ist, und ist daher eine Schranke für die Länge der Folgen in $\mathcal{F}$. Für jedes $F \in \mathcal{F}$ setzen wir

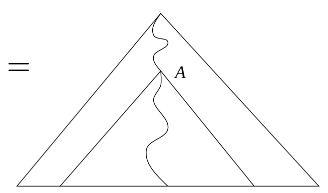
$$\begin{array}{lll} m^F & := & \sum_{i \leq k^F} \|Front(\tau_i^F)\| \\ \text{Baum}^F & := & \{\, \tau \mid \tau \text{ ist } A\text{-Baum von G für ein } A \in N,\ NT(\tau) \subseteq \bigcup_{i \leq k^F} NT(\tau_i^F) \,\} \\ S^F & := & \{\, m^F + \sum_{\tau \in \text{Baum}^F} n_\tau \cdot \|Front(\tau)\| \mid n_\tau \geq 0 \,\}. \end{array}$$

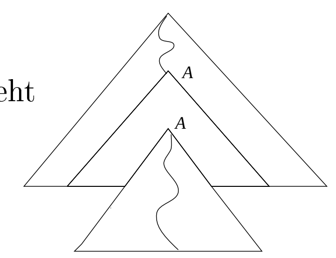
Offenbar ist S^F linear, und damit $\bigcup_{F \in \mathcal{F}} S^F$ semilinear. Es bleibt noch zu zeigen:

Behauptung: $\|L(G)\| = \bigcup_{F \in \mathcal{F}} S^F$.

Wir sagen, durch EINHÄNGEN von

$\tau_A =$ 

in $\tau =$ 

entsteht 

.

Beh. 1: $\bigcup_{F\in\mathcal{F}} S^F \subseteq \|L(G)\|$.

Beweis: Sei $F = (\tau_0^F, \ldots, \tau_{k^F}^F) \in \mathcal{F}$ und $v := (m^F + \sum_{\tau\in\mathrm{Baum}^F} n_\tau \cdot \|\mathit{Front}(\tau)\|) \ \in S^F$. Mit α) und β) kann man aus dem G-Ableitungsbaum τ_0^F mit Wurzel S durch sukzessives Einhängen von τ_1^F, τ_2^F usw. einen G-Ableitungsbaum τ^F konstruieren mit Wurzel S, $\mathit{Front}(\tau^F) \subseteq \Sigma^*$ und $||\mathit{Front}(\tau^F)|| = m^F$. Nun hängt man jeden Baum τ aus Baum^F n_τ-mal in τ^F ein, so daß man einen G-Ableitungsbaum ϱ mit $\mathit{Front}(\varrho) \in L(G)$ und

$$\|\mathit{Front}(\varrho)\| = m^F + \Sigma_{\tau\in\mathrm{Baum}^F}\ n_\tau \cdot \|\mathit{Front}(\tau)\| = v$$

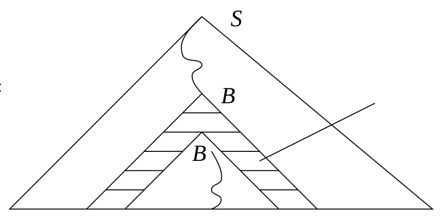
erhält. Also ist $v \in \|L(G)\|$ und daher $S^F \subseteq \|L(G)\|$.

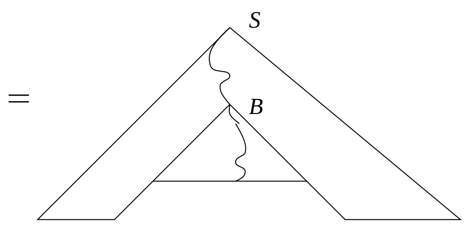
Beh. 2: $\|L(G)\| \subseteq \bigcup_{F\in\mathcal{F}} S^F$.

Beweis: Sei $w \in L(G)$ und ϱ ein Ableitungsbaum für w, also $\mathit{Front}(\varrho) = w$ und Wurzelmarke$(\varrho) = S$. Wir unterscheiden zwei Fälle.

(a) ϱ ist Basisbaum: Dann ist $F := (\varrho) \in \mathcal{F}$ und $\|w\| = m^F \in S^F$.

(b) ϱ ist kein Basisbaum: Nach Lemma 8.7 gibt es einen B-Baum τ_B mit

$\varrho =$ 

τ_B. Sei $\tau' =$ 

aus ϱ durch Weglassen von τ_B entstanden. Da $|\tau'| < |\varrho|$ ist, gibt es nach Induktion schon ein $F' = (\tau_0^{F'}, \ldots, \tau_{k^{F'}}^{F'}) \in \mathcal{F}$, so daß für geeignete $n'_\tau \geq 0$

$$\|\mathit{Front}(\tau')\| = m^{F'} + \sum_{\tau\in\mathrm{Baum}^{F'}} n'_\tau \cdot \|\mathit{Front}(\tau)\| \in S^{F'}.$$

Falls $\tau_B \in \mathrm{Baum}^{F'}$ ist, erhöhen wir n'_{τ_B} um 1 und erhalten

$$\|w\| = \|\mathit{Front}(\tau')\| + \|\mathit{Front}(\tau_B)\| \in S^{F'}.$$

Falls $\tau_B \notin \mathrm{Baum}^{F'}$ ist, setzen wir F' fort zu $F := (\tau_0^{F'}, \ldots, \tau_{k^{F'}}^{F'}, \tau_B)$. Da die Bedingungen a) – c) erfüllt sind, ist $F \in \mathcal{F}$ und damit

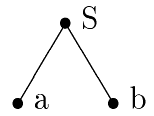
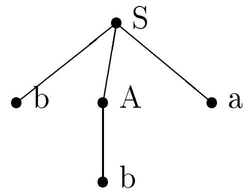
$$\|w\| = \|\mathit{Front}(\tau')\| + \|\mathit{Front}(\tau_B)\| = (m^F + \sum_{\tau\in\mathrm{Baum}^F} n_\tau \cdot \|\mathit{Front}(\tau)\|) \ \in S^F,$$

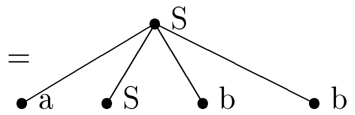
wobei $n_\tau = n'_\tau$ für $\tau \in \mathrm{Baum}^{F'}$ und $n_\tau = 0$ für $\tau \in \mathrm{Baum}^F - \mathrm{Baum}^{F'}$ gewählt wird.

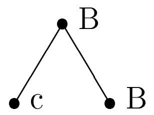
Damit ist $\|w\| \in S^F$ und folglich $\|L(G)\| \subseteq \bigcup_{F\in\mathcal{F}} S^F$. □

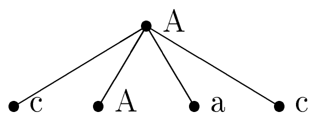
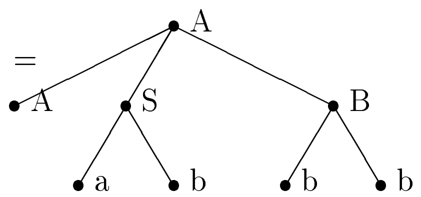
Beispiel 8.13 Wir bestimmen die Buchstabenhäufigkeit für die Grammatik G mit den Regeln

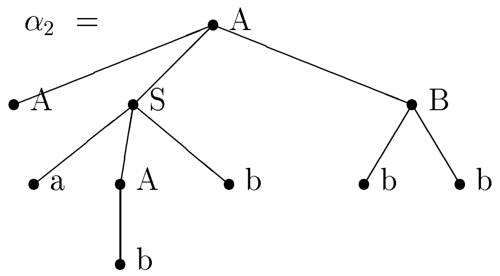
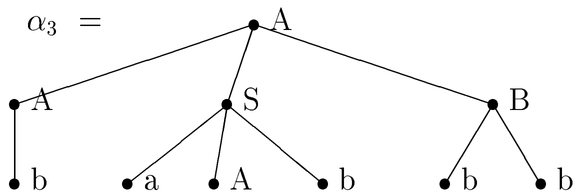
$$\begin{array}{lclclcl} S & \rightarrow & ab & \mid & aSbb & \mid & bAa \\ A & \rightarrow & b & \mid & cAac & \mid & ASB \\ B & \rightarrow & cB & \mid & bb & & \end{array}$$

Basisbäume mit Wurzel S: $\sigma_0 =$  $\sigma_1 =$ 

S-Bäume: $\sigma_2 =$ 

B-Bäume: $\beta_0 =$ 

A-Bäume: $\alpha_0 =$  $\alpha_1 =$ 

$\alpha_2 =$  $\alpha_3 =$ 

Zur Berechnung von $\|L(G)\|$ bestimmt man zuerst $\mathcal{F}$ und die Baummengen Baum^F zu $F \in \mathcal{F}$:

$$\begin{array}{lcll} F_1 & = & <\sigma_0>, & \mathrm{Baum}^{F_1} = \{\sigma_2\} \\ F_2 & = & <\sigma_1>, & \mathrm{Baum}^{F_2} = \{\sigma_2, \alpha_0\} \\ F_{2+i} & = & <\sigma_1, \alpha_i>, & \mathrm{Baum}^{F_{2+i}} = \{\sigma_2, \alpha_0, \alpha_1, \alpha_2, \alpha_3, \beta_0\}, \quad \text{für } i = 1,2,3. \end{array}$$

Mit $\|w\| = (|w|_a, |w|_b, |w|_c)$ erhält man

$$\begin{array}{lcll} S^{F_1} & = & \{(1,1,0) & +n\cdot(1,2,0) \mid n\in I\!N\} \\ S^{F_2} & = & \{(1,2,0) & +n\cdot(1,2,0)+m\cdot(1,0,2) \mid n,m\in I\!N\} \\ S^{F_{2+i}} & = & \{(1,2,0) & +\|Front(\alpha_i)\| \\ & & & +k_1\cdot(1,2,0)+k_2\cdot(1,0,2)+k_3\cdot(1,3,0) \\ & & & +k_4\cdot(1,4,0)+k_5\cdot(0,0,1) \mid k_1,\dots,k_5\in I\!N\}, \quad i=1,2,3. \end{array}$$

Daher ist $\|L(G)\| = S^{F_1}\cup\ldots\cup S^{F_5}$.

Korollar 8.14 (i) Zu jeder kontextfreien Sprache $L\subseteq\Sigma^*$ gibt es eine reguläre Sprache $R\subseteq\Sigma^*$ mit $\|R\| = \|L\|$.

(ii) Ist $|\Sigma| = 1$, so ist jede kontextfreie Sprache $L\subseteq\Sigma^*$ schon regulär.

Für Teil (ii) beachte man, daß für $\Sigma = \{a\}$ die Buchstabenhäufigkeit $|w|_a$ eines $w\in\Sigma^*$ mit w identifiziert werden kann, da $w = a^{|w|_a}$. Teil (i) folgt direkt aus dem Satz und bedeutet, daß sich bei Ignorieren der Wortstellung die kontextfreien Sprachen nicht mehr von den regulären unterscheiden.

Unter freier Wortstellung stellen wir uns aber etwas anderes als den Übergang zu den Häufigkeiten vor: in einer Sprache L mit freier Wortstellung sollten zu jedem $w\in L$ auch alle $v\in\Sigma^*$ mit $\|v\| = \|w\|$ zu L gehören.

Der Abschluss einer Sprache $L\subseteq\Sigma^*$ unter freier Wortstellung ist

$$\mathrm{perm}(L) := \{v\in\Sigma^* \mid \text{ Es gibt } w\in L \text{ mit } \|v\| = \|w\|\},$$

die kleinste Sprache $M\supseteq L$, die freie Wortstellung hat.

Die Klasse aller regulären Sprachen ist nicht unter freier Wortstellung abgeschlossen:

Beispiel 8.15 $\mathrm{perm}(L(ab)^*)$ ist kontextfrei, aber $\mathrm{perm}(L(abc)^*)$ ist nicht kontextfrei.

Die erste Behauptung zeigt man mit der kontextfreien Grammatik

$$S\to aSb \quad|\quad bSa \quad|\quad SS \quad|\quad \varepsilon$$

für $L(\mathrm{perm}((ab)^*))$. Für die zweite Behauptung benutzt man, daß

$$\mathrm{perm}(L(abc)^*)\cap L(a^*b^*c^*) = \{\, a^nb^nc^n \mid n\in I\!N\,\}$$

nach 8.10 nicht kontextfrei ist. Da für jedes kontextfreie L und reguläre R aber $L\cap R$ kontextfrei ist – vgl. Satz 8.16 –, kann $L(\mathrm{perm}((abc)^*))$ nicht kontextfrei sein.

Für natürliche Sprachen gilt sicher nicht, daß beliebige Sätze durch Umstellung von Wörtern wieder zu Sätzen bzw. Wörter durch Umstellen von Buchstaben wieder zu Wörtern der Sprache werden. Die Freiheit der Wortstellung gilt aber in manchen natürlichen Sprachen "lokal", das heißt in eingeschränktem Sinn. Um dies zu beschreiben, kann es genügen, kontextfreie Regeln $A\to A_1\ldots A_n$ mit gewissen Vertauschungen der A_i zulassen, was natürlich durch mehrere kontextfreie Regeln ausgedrückt werden kann.

8.4 Abschlußeigenschaften der Klasse kontextfreier Sprachen

Die Klasse der kontextfreien Sprachen hat bezüglich der Boole'schen Operationen schlechtere Abschlußeingenschaften als die der regulären Sprachen:

Satz 8.16 Sei CF_Σ die Menge der kontexfreien Sprachen über Σ. Dann gilt:

(i) $Lin_\Sigma \subset \bigcup_{m\in \mathbb{N}} m{-}Lin_\Sigma \subset CF_\Sigma$.

(ii) CF_Σ ist abgeschlossen unter den regulären Operationen $0, 1, a, +, \cdot, {}^*$ und unter Substitution. (Auch unter kleinsten Fixpunkten μ-regulärer Funktionen).

(iii) CF_Σ ist weder unter Durchschnitts- noch unter Komplementbildung abgeschlossen.

(iv) CF_Σ ist abgeschlossen unter Durchschnitt mit regulären Sprachen.

Beweis: (i) Da jede lineare Grammatik eine kontextfreie Grammatik ist und die Regeln $x_0 \to x_1 \cdots x_m$ ebenfalls kontextfrei sind, ist m-$Lin_\Sigma \subseteq CF_\Sigma$. Das reguläre Gleichungssystem

$$S = (Xc)^*X, \quad X = aXb + 1$$

definiert eine kontextfreie Sprache $L = (Ac)^*A$ mit der Hilfssprache $A = \{ a^n b^n \mid n \in \mathbb{N} \}$. Man kann zeigen, daß eine Sprache der Form $(Xc)^*X$ nur dann m-linear ist für irgend ein m, wenn X regulär ist. (Auf einen Beweis müssen wir hier verzichten, vgl. Harrison[10], Satz 6.8.2.) Das ist aber bei L nicht der Fall, da A nicht regulär ist.

(ii) Da die regulären Operationen beim Aufbau der μ-regulären Ausdrücke zugelassen sind, folgt der erste Teil aus Satz 4.23. Für den Abschluß unter Substitution benutzen wir, daß für μ-reguläre Ausdrücke $r, r_1, \ldots, r_k$ auch $r[r_1/a_1, \ldots, r_k/a_k]$ ein μ-regulärer Ausdruck ist. Durch Induktion erhält man $L(r[r_1/a_1, \ldots, r_k/a_k]) = L(r)[L(r_1)/a_1, \ldots, L(r_k)/a_k]$.

(iii) Sei $L_1 = \{a^i b^j c^j \mid i, j \in \mathbb{N}\}$, und $L_2 = \{a^i b^i c^k \mid i, k \in \mathbb{N}\}$. Dann ist $L_1 = L(a^*)\cdot\{b^j c^j \mid j \in \mathbb{N}\}$ das Produkt zweier kontextfreier Sprachen, also selbst kontextfrei. Ebenso für L_2. Aber $L_1 \cap L_2 = \{a^n b^n c^n \mid n \in \mathbb{N}\}$ ist nach 8.10 nicht kontextfrei. Wäre mit L_i auch $\overline{L_i} := \Sigma^* - L_i$ kontextfrei, so wäre auch $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$ kontextfrei, im Widerspruch zum gerade Gezeigten.

(iv) Sei L eine kontextfreie und R eine reguläre Sprache. L werde durch den rekursiven endlichen Automaten $\mathcal{A} = (Q^A, \vdash_a^A, \vdash_{call}^A, \vdash_{return}^A, q_0^A, F^A)_{a\in\Sigma}$ und R durch den endlichen Automaten $\mathcal{B} = (Q, \vdash_a, q_0, F)_{a\in\Sigma}$ erkannt. Wir konstruieren einen rekursiven endlichen Automaten $\mathcal{A}''$, der $L \cap R$ erkennt.

Zunächst machen wir $\mathcal{B} = (Q, \vdash_a, q_0, F)_{a\in\Sigma}$ zu einem rekursiven endlichen Automaten $\mathcal{B}' = (Q \times Q, \vdash'_a, \vdash'_{call}, \vdash'_{return}, (q_0, q_0), F \times \{q_0\})$ durch

$$\begin{array}{lll}(p,p') \vdash'_a (q,q') & :\iff & p \vdash_a q \ \wedge \ p' = q' \\ (p,p') \vdash'_{call} (r,r')(q,q') & :\iff & q = p \ \wedge \ r = q' \ \wedge \ r' = p', \\ (p,p') \vdash'_{return} \varepsilon & :\iff & p = p'.\end{array}$$

Bei einem rekursiven Aufruf im Zustand (p,p') –mit $\mathcal{B}$ im Zustand p und Ziel p'– rät man, in welchem Zustand q' von $\mathcal{B}$ die Bearbeitung des Aufrufs enden wird, und speichert sich diesen in der zweiten Komponente von $(q,q') = (p,q')$. Hat $\mathcal{B}$ das neue Ziel q' erreicht, kann der Rekursionsaufruf beendet werden. Dann sollte $\mathcal{B}'$ die Berechnung mit Ziel p' fortsetzen, d.h. die Rücksprungadresse ist $(r,r') = (q',p')$.

Durch Induktion über die Schachtelungstiefe n eines durch *call*-*return*-Paare 'wohlgeklammerten' Wortes $w \in (\Sigma \cup \{call, return\})^*$ beweist man

$$(p,p') \vdash'_w (q,q') \implies p \vdash_{erase(w)} q \ \wedge \ q' = p'.$$

Das ist für $n = 0$ offensichtlich, und für $n > 0$ zurückführbar auf $p \vdash_{erase(v)} q'$ in

$$(p,p') \ \vdash'_{call} \ (q',p')\,(p,q') \ \vdash'_v \ (q',p')\,(q',q') \ \vdash'_{return} \ (q',p').$$

Damit ist für $erase(w) \in L(\mathcal{B}')$ wegen $(q_0,q_0) \vdash'_w (q,q_0) \in F \times \{q_0\}$ auch $q_0 \vdash_{erase(w)} q \in F$, also $L(\mathcal{B}') \subseteq L(\mathcal{B})$. Da für $q_0 \vdash_w q \in F$ auch $(q_0,q_0) \vdash'_v (q,q_0)$ gilt, ist $L(\mathcal{B}) \subseteq L(\mathcal{B}')$.

Nun lassen wir $\mathcal{A}$ und $\mathcal{B}'$ parallel arbeiten, wobei wir rekursive Aufrufe und Rücksprünge von $\mathcal{B}'$ mit denen von $\mathcal{A}$ synchronisieren.[6] Sei

$$\mathcal{A}'' = (Q^A \times Q \times Q, \vdash''_a, \vdash''_{call}, \vdash''_{return}, (q_0^A, q_0, q_0), \ F^A \times F \times \{q_0\})$$

der rekursive endliche Automat mit den Übergangsrelationen

$$\begin{array}{lllll}(p^A,p,p') \vdash''_a (q^A,q,q') & :\iff & p^A \vdash^A_a q^A & \wedge & (p,p') \vdash'_a (q,q'), \\ (p^A,p,p') \vdash''_{call} (r^A,r,r')(q^A,q,q') & :\iff & p^A \vdash^A_{call} r^A\, q^A & \wedge & (p,p') \vdash'_{call} (r,r')\,(q,q'), \\ (p^A,p,p') \vdash''_{return} \varepsilon & :\iff & p^A \vdash^A_{return} \varepsilon & \wedge & (p,p') \vdash'_{return} \varepsilon.\end{array}$$

Damit ist $L(\mathcal{A}'') = L \cap R$: Denn es ist $(q_0^A, q_0, q_0) \vdash''_w (q^A, q, q') \in F^A \times F \times \{q_0\}$ genau dann der Fall, wenn $q_0^A \vdash^A_w q^A \in F^A$ und $(q_0,q_0) \vdash'_w (q,q_0) \in F \times \{q_0\}$, also $erase(w) \in L(\mathcal{A}'')$ genau dann, wenn $erase(w) \in L(\mathcal{A}) = L$ und $erase(w) \in L(\mathcal{B}') = L(\mathcal{B}) = R$. □

Nimmt man auch für natürliche Sprachen an, daß syntaktische Konstruktionen beliebig tief geschachtelt aufteten dürfen, so erhält man die – keineswegs offensichtliche –

Folgerung 8.17 (Shieber 1987) Es gibt natürliche Sprachen, die nicht kontextfrei sind.

[6] Für beliebige rekursive endliche Automaten statt $\mathcal{B}'$ geht das nicht, so daß die Konstruktion nicht zur Erkennung des Durchschnitts zweier kontextfreier Mengen taugt.

Beweis: Die entsprechende Behauptung dafür, daß natürliche Sprachen i.a. nicht regulär sind, kann auf Schachtelungen ("nested dependencies")

(The mouse) (the cat) (the dog) bit chased died.

beliebiger Tiefe zurückgeführt werden, d.h. darauf, daß

$$\{a_1 \dots a_m a_m \dots a_1 \mid m \in I\!N, a_i \in \Sigma\}$$

bzw. noch einfacher, daß $\{a^m b^m \mid m \in I\!N\}$ nicht regulär ist. Für den kontextfreien Fall betrachten wir sich überkreuzende Abhängigkeiten ("cross-serial-dependencies") der Form

$$\mathrm{NP}^n_{\mathrm{akk}} \ \mathrm{NP}^m_{\mathrm{dat}} \qquad \mathrm{V}^n[\mathrm{akk}] \ \mathrm{V}^m[\mathrm{dat}]$$

Hierbei steht $\mathrm{NP}_{\mathrm{akk}}$ für Nominalphrasen im Akkusativ und V[akk] für die Verben, die ein (direktes) Objekt im Akkusativ erfordern. Als Beispiele, in denen diese Art der Abhängigkeit in natürlichsprachlichen Sätzen vorkommt, gibt Shieber folgende Sätze aus dem Berner Dialekt des Schwyzerdeutschen an:

$$\sigma_{n,m} = \begin{cases} \text{Jan säit das mer (d'chind)}^n \text{ (em Hans)}^m \text{ es huus} \\ \text{haend wele (laa)}^n \text{ (hälfe)}^m \text{ aastriche.} \end{cases}$$

Beachte, daß die Klammerung im Hochdeutschen anders ist: Jan sagt, wir hätten (die $\text{Kinder}_{\text{akk}}$ (dem Hans_{dat} (das Haus_{akk} anstreichen[akk]) helfen[dat]) lassen[akk]) wollen.) Man führt das Problem also darauf zurück, daß

$$L := \{ua^n b^m vc^n d^m w \mid n, m \geq 0\}$$

nicht kontextfrei ist, für feste u, v, w. Das wiederum zeigt man mit einer Verallgemeinerung des Wiederholungslemmas, was hier übergangen wird. Sei nun R die von dem regulären Ausdruck

$$r := \begin{cases} \text{Jan säit das mer (d'chind)}^* \text{ (em Hans)}^* \text{ es huus} \\ \text{haend wele (laa)}^* \text{ (hälfe)}^* \text{ aastriche} \end{cases}$$

definerte reguläre Sprache über einem geeignetem Alphabet Σ. Dann ist

$$R \ \cap \ \text{Schwyzerdeutsch} \ = \{\sigma_{n,m} \mid n, m \in I\!N\}$$

nicht kontextfrei, also nach Teil (iv) von Satz 8.16 auch Schwyzerdeutsch nicht. $\square$

8.5 Entscheidbarkeitseigenschaften

Satz 8.18 (i) Es gibt Verfahren, mit denen man für kontextfreie Grammatiken G mit Terminalalphabet Σ, Wörter $w \in \Sigma^*$ und $m \in I\!N$ feststellen kann, welche der folgenden Aussagen gelten:

i) $w \in L(G)$ ii) $L(G) = \emptyset$ iii) $|L(G)| < m$ iv) $|L(G)| < \infty$.

(ii) Es gibt keine Verfahren, mit denen man für alle kontextfreien Grammatiken G_i mit Terminalalphabet Σ, $|\Sigma| > 1$, und alle regulären Ausdrücke r über Σ feststellen könnte, welche der folgenden Aussagen gelten:

i) $L(G_1) = \Sigma^*$, ii) $L(G_1) = L(r)$, iii) $L(G_1)$ ist regulär,
iv) $L(G_1) \subseteq L(G_2)$ v) $L(G_1) = L(G_2)$.

Beweis: i) Die Frage, ob $w \in L(G)$ liegt, kann mit den Verfahren zur Syntaxanalyse in Kapitel 9 beantwortet werden.

Aus Lemma 8.7 und 8.8 folgt, daß auch die anderen Fragen entscheidbar sind, da man die Basisbäume und alle A-Bäume von G effektiv ermitteln und daran die Antworten ablesen kann.

ii) Die Unentscheidbarkeitsaussagen können wir hier nicht beweisen, da wir die dazu nötigen Hilfsmittel, wie man über alle möglichen Verfahren etwas zeigt, nicht behandelt haben. Man beachte aber, daß die Unentscheidbarkeit der Aussagen in ii), iv) und v) aus der in i) folgt. □

Die Unentscheidbarkeitseigenschaften gelten auch schon für die Teilklasse der linearen Grammatiken (vgl. Abschnitt 11).

Klammergrammatiken und starke Äquivalenz

Während die Frage, ob zwei kontextfreie Grammatiken G_1 und G_2 SCHWACH ÄQUIVALENT sind, d.h. ob $L(G_1) = L(G_2)$ gilt, unentscheidbar ist, kann man die entsprechende Frage für die STARKE ÄQUIVALENZ entscheiden.

Bei der starken Äquivalenz zweier Grammatiken verlangt man nicht nur die Gleichheit der erzeugten Wortmengen, sondern auch, daß dieselben Worte in gleicher Weise syntaktisch strukturiert werden. In Programmiersprachen und logischen Sprachen macht man die Struktur durch die Verwendung von Klammern sichtbar.

Definition 8.19 Eine KLAMMERGRAMMATIK ist eine kontextfreie Grammatik G mit einem Paar $\{(,)\} \subseteq \Sigma$ von KLAMMERSYMBOLEN, bei der alle Regeln die Form $A \to (\alpha)$ haben, wobei α keines der Klammersymbole enthält.

Satz 8.20 (McNaughton 1967) Es ist entscheidbar, ob zwei Klammergrammatiken äquivalent sind.

Beweis: Man konstruiert aus einer Klammergrammatik G eine äquivalente Klammergrammatik G' in geeigneter Normalform, für die die Äquivalenz dazu gleichwertig ist, daß es eine Isomophie zwischen den syntaktischen Kategorien gibt, die die Regeln respektiert. Wir brauchen mehrere Lemmata.

Eine kontextfreie Grammatik G heißt RÜCKWÄRTS DETERMINISTISCH, falls die linke Regelseite ihrer Regeln durch die rechte eindeutig bestimmt ist, d.h. wenn für Regeln $A \to \alpha$ und $B \to \alpha$ von G stets $A \equiv B$ gilt.

Lemma 8.21 Zu jeder Klammergrammatik G gibt es eine rückwärts deterministische Klammergrammatik G' mit $L(G) = L(G')$.

Beweis: Zu $G = (N, \Sigma, P, S)$ sei $G' := (N', \Sigma, P', S')$ mit $N' := 2^N$, $S' := \{B \subseteq N \mid S \in B\}$ und für alle $B_i \in N'$ und $w_i \in \Sigma^*$ sei $B \to (w_1 B_1 \cdots w_k B_k w_{k+1}) \in P'$ genau dann wenn

$$B = \{A \in N \mid \exists A_1 \in B_1 \ldots \exists A_k \in B_k \;(A \to (w_1 A_1 \cdots w_k A_k w_{k+1}) \in P)\}.$$

Da offensichtlich in jeder Regel von P' die linke Seite durch die rechte eindeutig bestimmt ist, ist G' rückwärts deterministisch.

Beh. 1: $L(G) \subseteq L(G')$: Ersetze in einer G-Ableitung vom Ende her alle Regelanwendungen durch Regelanwendungen in G'. Jeder Regel $A \to w \in P$ mit $w \in \Sigma^*$ entspricht eine Regel $B \to w$ mit $A \in B$ von G'. Hat man schon einer Satzform α von G eine Satzform β von G' zugeordnet, so findet man zu einer rechten Regelseite in α eine zugeordnete rechte Regelseite in β. Die linke Seite B der G'-Regel $B \to \beta$ ist eindeutig bestimmt und erfüllt $A \in B$ für die rechte Seite A der zur Herleitung von α benutzten G-Regel $A \to \alpha$.

Beh. 2: $L(G') \subseteq L(G)$: Ersetze in einer G'-Ableitung vom Anfang her jede Anwendung einer G'-Regel $B_0 \to (w_1 B_1 \cdots w_k B_k w_{k+1})$ durch eine passende G-Regel

$$A_0 \to (w_1 A_1 \cdots w_k A_k w_{k+1})$$

mit $A_i \in B_i$ für alle i. Dies geht wegen der Wahl von S' und den Regeln von G'. □

Wie in Proposition 6.29 können wir zusätzlich erreichen, daß G' keine überflüssigen Symbole enthält.

Proposition 8.22 Ist eine G rückwärts deterministische Klammergrammatik ohne überflüssige Symbole, so ist $L(A) \cap L(B) = \emptyset$ für Nonterminale $A \not\equiv B$.

Beweis: Da G keine überflüssigen Symbole hat, ist $L(A) \neq \emptyset$ für jedes Nonterminal A. Da G eine Klammergrammatik ist, gibt es zu $w \in L(A)$ nur einen Analysebaum: weil alle Regeln die Form $X \to (\alpha)$ haben, worin α keine Klammern enthält, ordnet jeder Analysebaum von w einem Teilwort (v) von w mit klammerfreiem v eine nur von v abhängende Teilanalyse $B_v \to (v)$ zu. Entsprechend argumentiert man für die Satzform α, die man aus w erhält, indem man alle Teilworte (v) mit klammerfreiem v durch ihre Kategorien B_v ersetzt. Also hat w nur einen Analysebaum, und seine Wurzelkategorie ist eindeutig bestimmt. Es kann also kein $w \in L(A) \cap L(B)$ mit $A \not\equiv B$ geben. $\square$

Definition 8.23 Sei $G = (N, \Sigma, P, S)$ eine kontextfreie Grammatik. Ein nicht-terminaler KONTEXT $\alpha_1 \square \alpha_2$ von G besteht aus einem Wort $\alpha_1\alpha_2 \in (N \cup \Sigma)^*$ mit einer Lücke $\square \notin N \cup \Sigma$. Zwei Nonterminale $A, B \in N$ sind KONTEXT-UNUNTERSCHEIDBAR, falls

$$A \simeq_{K(G)} B \quad :\Longleftrightarrow \quad \forall \alpha_1, \alpha_2 \in (N \cup \Sigma)^* (S \Longrightarrow_G^* \alpha_1 A \alpha_2 \iff S \Longrightarrow_G^* \alpha_1 B \alpha_2).$$

Eine Menge $M \subseteq N$ heißt n-KONTEXTDEFINIERBAR IN G, falls

$$M = \{ A \in N \mid S \Longrightarrow_G^{\leq n} \alpha_1 A \alpha_2 \}$$

für einen Kontext $\alpha_1 \square \alpha_2$ von G.

Lemma 8.24 Für rückwärts deterministische Klammergrammatiken $G = (N, \Sigma, P, S)$ und $M \subseteq N$ ist effektiv feststellbar, ob M in G n-kontextdefinierbar ist für ein $n \in \mathbb{N}$.

Beweis: Man zeigt zunächst, daß nur die $2^{|N|}$-Kontextdefinierbarkeit in Frage kommt. Da aber durch die Ableitungslänge auch die Anzahl der möglichen Kontexte beschränkt ist, folgt daraus die Behauptung.

Beh.: Hat G eine $n+2$-kontextdefinierbare Menge, die nicht $n+1$-kontextdefinierbar ist, so hat G eine $n+1$-kontextdefinierbare Menge, die nicht n-kontextdefinierbar ist.

Bew: Angenommen, $M = \{ A \in N \mid S \Longrightarrow_G^{\leq n+2} \alpha_1 A \alpha_2 \}$ ist nicht $(n+1)$-kontextdefinierbar. Dann gibt es $A \in M$ mit

$$S \Longrightarrow_G^{n+2} \alpha_1 A \alpha_2,$$

und nach der Form der Regeln hat $\alpha_1\alpha_2$ genau $n+2$ Klammerpaare; daher ist

$$M = \{ A \in N \mid S \Longrightarrow_G^{n+2} \alpha_1 A \alpha_2 \}.$$

Sei $\alpha_1 \square \alpha_2 \equiv \beta_1(\gamma_1 \square \gamma_2)\beta_2$, wobei $(\gamma_1 \square \gamma_2)$ ein minimaler wohlgeklammerter Teilausdruck von $\alpha_1 \square \alpha_2$ ist, der die Lücke $\square$ enthält. Dann sind γ_1 und γ_2 klammerfrei. Sonst wäre etwa $\gamma_1 \equiv (\gamma_{1,1}(\gamma)\gamma_{1,2})$ mit klammerfreiem γ, und es gäbe genau ein $C \in N$ mit $C \to (\gamma) \in P$. Mit $\tilde{\gamma}_1 := \gamma_{1,1} C \gamma_{1,2}$ wäre dann

$$M = \{ A \in N \mid S \Longrightarrow_G^{n+1} \beta_1(\tilde{\gamma}_1 A \gamma_2)\beta_2 \},$$

eine $n+1$-definierbare Menge, im Widerspruch zur Annahme. Da nun γ_1, γ_2 klammerfrei sind, hat $\beta_1\Box\beta_2$ genau $n+1$ Klammerpaare, und

$$M' := \{\, B \in N \mid S \overset{n+1}{\underset{G}{\Longrightarrow}} \beta_1 B \beta_2 \,\}$$

ist $n+1$-kontextdefierbar. M' ist aber nicht n-kontextdefinierbar, denn sonst wäre für einen geeigneten Kontext $\tilde\beta_1\Box\tilde\beta_2$

$$\begin{aligned} M &= \{\, A \in N \mid \text{Es gibt } B \in M' \ (S \overset{n+1}{\underset{G}{\Longrightarrow}} \beta_1 B \beta_2 \overset{1}{\underset{G}{\Longrightarrow}} \beta_1(\gamma_1 A \gamma_2)\beta_2) \,\} \\ &= \{\, A \in N \mid \text{Es gibt } B \in M' \ (S \overset{n}{\underset{G}{\Longrightarrow}} \tilde\beta_1 B \tilde\beta_2 \overset{1}{\underset{G}{\Longrightarrow}} \tilde\beta_1(\gamma_1 A \gamma_2)\tilde\beta_2) \,\} \end{aligned}$$

eine $(n+1)$-kontextdefinierbare Menge.

Wenn es eine $n+2$-kontextdefinierbare, nicht $n+1$-kontextdefinierbare Menge $M \subseteq N$ gibt, so gibt es nach der Behauptung mindestens $n+2$ verschiedene kontextdefinierbare Mengen. Daher ist jede der höchstens $2^{|N|}$ vielen kontextdefinierbaren Mengen schon n-kontextdefinierbar für ein $n \leq 2^{|N|}$. □

Korollar 8.25 Ist G eine rückwärts deterministische Klammergrammatik, so ist

$$A \simeq_{K(G)} B \iff \text{Für jede } 2^{|N|}\text{-kontextdefinierbare Menge } M\ (A \in M \leftrightarrow B \in M).$$

Insbesondere ist $A \simeq_{K(G)} B$ eine effektiv feststellbare Beziehung zwischen Nonterminalen.

Beweis: Ist $\alpha_1\Box\alpha_2$ ein Kontext mit $S \overset{*}{\underset{G}{\Longrightarrow}} \alpha_1 A \alpha_2$ und $S \overset{*}{\underset{G}{\not\Longrightarrow}} \alpha_1 B \alpha_2$, so ist

$$M := \{\, C \in N \mid S \overset{*}{\underset{G}{\Longrightarrow}} \alpha_1 C \alpha_2 \,\} = \{\, C \in N \mid S \overset{n}{\underset{G}{\Longrightarrow}} \alpha_1 C \alpha_2 \,\}$$

eine n-kontextdefinierbare Menge mit $A \in M$ und $B \notin M$, wobei n die Anzahl der Klammerpaare in $\alpha_1\Box\alpha_2$ ist. Nach dem Lemma ist M schon $2^{|N|}$-kontextdefinierbar. Daraus folgt die Behauptung. □

Eine Klammergrammatik $G = (N, \Sigma, P, S)$ heißt REDUZIERT, falls sie keine überflüssigen Nonterminale hat und $A \not\simeq_{K(G)} B$ für alle $A \not\equiv B \in N$ ist.

Lemma 8.26 Zu jeder rückwärts deterministischen Klammergrammatik G gibt es eine reduzierte rückwärts deterministische Klammergrammatik G' mit $L(G') = L(G)$.

Beweis: Wir können annehmen, daß G keine überflüssigen Symbole hat. Dann ist jedes Nonterminal von S erreichbar, liegt also in einer kontextdefinierbaren Menge $M \subseteq N$. Nach 8.25 kann man die $\simeq_{K(G)}$-Äquivalenzklasse

$$[A] := \{\, B \in N \mid A \simeq_{K(G)} B \,\}$$

von $A \in N$ effektiv berechnen. Für $\alpha \in (N \cup \Sigma)^*$ definiere $\alpha' \in (N' \cup \Sigma)^*$ rekursiv durch

$$\alpha' := \begin{cases} [A]\beta', & \text{falls } \alpha \equiv A\beta,\ A \in N, \\ a\beta', & \text{falls } \alpha \equiv a\beta,\ a \in \Sigma, \\ \varepsilon, & \text{falls } \alpha \equiv \varepsilon. \end{cases}$$

Sei $G' := (N', \Sigma, P', S')$ mit $N' := \{\, [A] \mid A \in N \,\}$ und $P' := \{\, A' \to \alpha' \mid A \to \alpha \in P \,\}$. Dann gilt für alle $\alpha \in (N \cup \Sigma)^*$:

$$S \overset{*}{\underset{G}{\Longrightarrow}} \alpha \quad \Longleftrightarrow \quad S' \overset{*}{\underset{G'}{\Longrightarrow}} \alpha'.$$

Wir zeigen nur $\Leftarrow$ durch Induktion über n in $S' \overset{n}{\underset{G'}{\Longrightarrow}} \alpha'$. Für $n = 0$ ist $\alpha' \equiv S'$ und $|\alpha| = |\alpha'| = 1$. Daher ist $\alpha \in N$ und $S \overset{*}{\underset{G}{\Longrightarrow}} \alpha$ wegen $S \overset{*}{\underset{G}{\Longrightarrow}} S$ und $[\alpha] = [S]$.

Für $n+1$ sei $S' \overset{n}{\underset{G'}{\Longrightarrow}} \alpha'_1 A' \alpha'_2 \overset{1}{\underset{G'}{\Longrightarrow}} \alpha'_1(\alpha')\alpha'_2$. Es gibt $B \to (\beta) \in P$ mit $A' \equiv B'$ und $\beta' \equiv \alpha'$. Wegen $S' \overset{n}{\underset{G'}{\Longrightarrow}} \alpha'_1(B')\alpha'_2$ ist nach Induktion $S \overset{*}{\underset{G}{\Longrightarrow}} \alpha_1 B \alpha_2$, daher auch $S \overset{*}{\underset{G}{\Longrightarrow}} \alpha_1(\beta)\alpha_2$. Daraus folgt aber $S \overset{*}{\underset{G}{\Longrightarrow}} \alpha_1(\alpha)\alpha_2$, denn wenn etwa $\alpha = C\gamma$ und $\beta = D\delta$ ist, so ist $\gamma' \equiv \delta'$ und wegen $C \simeq_{K(G)} D$ auch $S \overset{*}{\underset{G}{\Longrightarrow}} \alpha_1(C\delta)\alpha_2$. Entsprechend verfährt man mit γ und δ.

Daß G' rückwärts deterministisch und reduziert ist, bleibt dem Leser überlassen. □

Lemma 8.27 Zwei reduzierte rückwärts deterministische Klammergrammatiken G_1, G_2 mit $L(G_1) = L(G_2)$ sind isomorph.

Beweis: Sei $A \in N_1$. Da A nicht überflüssig ist, gibt es eine Ableitung

$$S_1 \overset{*}{\underset{G_1}{\Longrightarrow}} (\alpha_1 A \alpha_2) \overset{*}{\underset{G_1}{\Longrightarrow}} (w_1(w)w_2) \ \in \Sigma^*.$$

Wegen $(w_1(w)w_2) \in L(G_2)$ gibt es $A' \in N_2$ mit $A' \overset{*}{\underset{G_2}{\Longrightarrow}} (w)$. Angenommen, es gibt $(v) \in L(A) - L(A')$. Mit Proposition 8.22 folgt, daß für jeden Kontext $u_1 \Box u_2$ mit $u_1 u_2 \in \Sigma^*$(!)

$$S_1 \overset{*}{\underset{G_1}{\Longrightarrow}} u_1(w)u_2 \quad \Longleftrightarrow \quad u_1 L(A) u_2 \subseteq L(G_1) = L(G_2).$$

Wie für (w) gibt es $B' \in N'$ mit $B' \overset{*}{\underset{G_2}{\Longrightarrow}} (v)$, und wieder nach 8.22 ist

$$S_2 \overset{*}{\underset{G_2}{\Longrightarrow}} u_1 A' u_2 \quad \Longleftrightarrow \quad S_2 \overset{*}{\underset{G_2}{\Longrightarrow}} u_1 B' u_2 \tag{52}$$

für alle Kontexte $u_1 \Box u_2$ mit $u_1, u_2 \in \Sigma^*$. Da G_2 keine überflüssigen Symbole hat, gilt (52) auch für alle nicht-terminalen Kontexte, weshalb $A' \simeq_{K(G_2)} B'$ ist. Da G_2 reduziert ist, müßte $A' \equiv B'$ sein, im Widerspruch zu $(v) \in L(B') - L(A')$. Es folgt, daß $L(A) \subseteq L(A')$ und analog auch $L(A') \subseteq L(A)$ ist.

Also gibt es zu jedem $A \in N_1$ ein $A' \in N_2$ mit $L(A) = L(A')$. Beachte, daß $S_1' \equiv S_2$ sein muß. Es bleibt zu zeigen, daß diese Zuordnung die Regeln von P_1 in die von P_2 überführt. Sei $A \to (aBbbC)$ eine Regel aus P_1, und $A', B', C' \in N_2$ mit $L(A') = L(A), L(B') = L(B)$ und $L(C') = L(C)$. Da die Grammatiken keine überflüssigen Symbole haben, gibt es $u \in L(B)$, $v \in L(C)$ und nach dem Beweis von 8.22 hat $(aubbv)$ in G_2 nur einen Analysebaum; er verzweigt an der Baumwurzel gemäß $A' \to (aB'bbC')$. Also ist die Übersetzung einer Regel von P_1 in P_2. Aus Symmetriegründen gilt auch die Umkehrung. □

Damit können wir den Beweis des Satzes von McNaughton abschließen: zu G_1 und G_2 konstruieren wir nach Lemma 8.21 und 8.26 zwei reduzierte rückwärts deterministische Klammergrammatiken G_1' und G_2' mit $L(G_1') = L(G_1')$ und $L(G_2') = L(G_2')$. Nun testen wir, ob es eine Bijektion zwischen den Nonterminalmengen von G_1' und G_2' gibt, die die Startsymbole und Regeln in einander überführt. Nach Lemma 8.27 ist die Antwort hierauf auch die Antwort auf die Ausgangsfrage. □

Die Aussage des Satzes gilt auch für gewisse Grammatiken (z.B. die "polnische Notation" der Formeln der Aussagenlogik), die keine Klammersymbole haben, sofern es

(i) keine verschiedenen Regeln $A \to \alpha\beta$, $B \to \beta\gamma$ mit $\beta \not\equiv \varepsilon$,

(ii) keine verschiedenen Regeln $A \to \alpha\beta\gamma$, $B \to \beta$, und

(iii) keine Regel $A \to \alpha\beta\alpha$ mit $\alpha \not\equiv \varepsilon$

gibt. Durch diese Abschwächung von "rückwärts deterministisch" wird noch sichergestellt, daß kein $\alpha \in (N \cup \Sigma)^*$ Analysen mit überlappenden Konstituenten hat.

Für die Analyse natürlicher Sprachen beachte man, daß die Anfangs- und Endgrenzen von Teilausdrücken oft sehr deutlich markiert sind, z.B. im Deutschen durch Kommata (bzw. Intonation), unterordnende Konjunktionen und Endstellung des finiten Verbs.

Aufgaben

Aufgabe 8.1 Sei $G = (N, \Sigma, P, \mathrm{NP})$ folgende Grammatik für Nominalphrasen im Plural:

N = {NP, N, Rel, VP, V}

Σ = { die, und, Hunde, Katzen, Mäuse, beißen, jagen, "," }

P bestehe aus den Regeln

1: NP → die N	6: N → Mäuse
2: NP → NP und NP	7: Rel → die VP
3: NP → NP, Rel,	8: VP → NP V
4: N → Hunde	9: V → beißen
5: N → Katzen	10: V → jagen

(i) Man gebe eine Ableitung und den zugehörigen Analysebaum für den Ausdruck *die Hunde und die Katzen, die die Mäuse jagen,* an.

(ii) Man gebe einen zweiten Analysebaum an, der zeigt, daß der Ausdruck syntaktisch mehrdeutig ist.

(iii) Man konstruiere nach dem Beweis aus der Vorlesung eine äquivalente Grammatik ohne Linksrekursion.

Aufgabe 8.2 Die Formeln der Prädikatenlogik erster Stufe mit einem zweistelligen Relationszeichen R und einem zweistelligen Funktionszeichen f werden durch die Grammatik mit den Regeln

$$\begin{array}{lcl} V & \to & x \mid V' \\ T & \to & V \mid f(T,T) \\ Q & \to & \exists \mid \forall \\ F & \to & R(T,T) \mid \neg F \mid QVF \mid (F \wedge F) \mid (F \vee F) \end{array}$$

und dem Startsymbol F erzeugt. Dabei stehen T, V, Q, F für die syntaktischen Kategorien der Variablen, Terme, Quantoren und Formeln. Die Variablen x, x', x'', $\ldots$ werden üblicherweise mit x_0, x_1, x_2, usw. bezeichnet. Als Terminalalphabet wird hier

$$\{R, f, x, (,), \text{``,''}, \text{``}{\prime}\text{''}, \neg, \vee, \wedge, \exists, \forall\}$$

verwendet.

a) Man gebe alle Basisbäume und alle A-Bäume für die verschiedenen Kategorien A der Grammatik an.

b) Man gebe das kommutative Bild $\| L \|$ der durch die Grammatik definierten (Teil-)Sprache $L(T)$ der Terme an. (Vergleiche mit dem Beispiel aus der Vorlesung.)

c) Einfachheitshalber kann man statt der prädikatenlogischen Formeln in a) und b) auch die aussagenlogischen Formeln in den beiden Aussagenvariablen p und q nehmen, die durch die Grammatik

$$F \;\to\; p \mid q \mid \neg F \mid (F \;\wedge\; F)$$

definiert werden. Dabei ist F das Startsymbol und $\{\neg,\ \wedge,\ (,\ ),\ p,\ q\}$ die Menge der Terminalsymbole. Man bestimme das kommutative Bild $\| AL \|$ der aussagenlogischen Formeln $AL = L(F)$.

Aufgabe 8.3 Die Klasse der kontextfreien Sprachen ist abgeschlossen unter "iterierter Substitution", d.h. kleinsten Fixpunkten regulärer Funktionen: ist $r(x_0, \ldots, x_n)$ ein μ–regulärer Ausdruck, so ist für kontextfreie Sprachen $L_1, \ldots, L_n$ auch $(\mu x_0.r)^{\mathcal{L}}(L_1, \ldots, L_n)$ eine kontextfreie Sprache.

9 Syntaxanalyse kontextfreier Sprachen

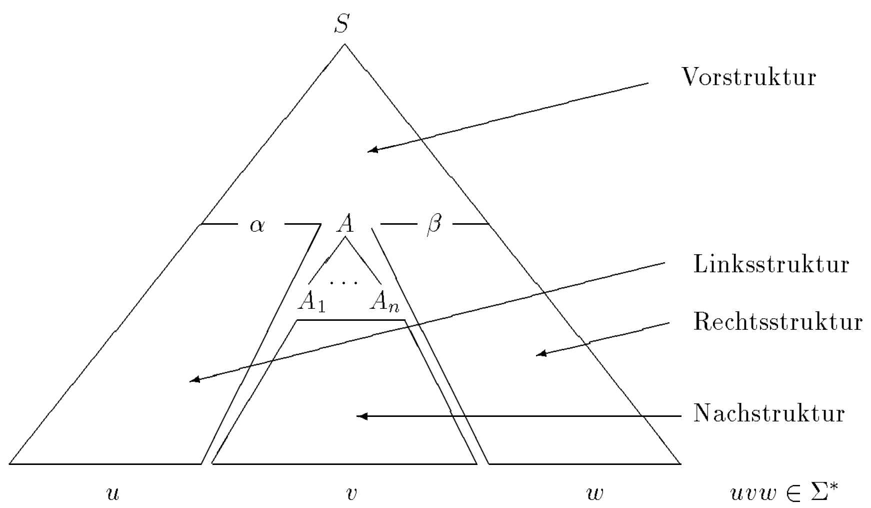
Die Aufgabe der Syntaxanalyse besteht darin, zu einer gegebenen Sprache $L \subseteq \Sigma^*$ und einem Wort $w \in \Sigma^*$ zu erkennen, ob $w \in L$ gilt oder nicht. Falls $w \in L$ ist, sollte die Analyse das dadurch nachweisen, daß sie alle (oder mindestens eine) der "syntaktischen Strukturen von w" konstruiert.

Als bloße Teil*menge* $L \subseteq \Sigma^*$ kann L nicht festlegen, was eine syntaktische Struktur ist[7]. Erst eine konstruktive Beschreibung von L, die angibt, wie L aus einfacheren Sprachen aufgebaut ist, legt syntaktische Strukturen für $w \in L$ fest: diese beschreiben, wie w gemäß der Konstruktuion von L entsteht.

Phrasenstrukturgrammatiken sind die üblichen "konstruktiven Beschreibungen" von Sprachen. Für kontextfreie Grammatiken G sind die G-Ableitungsbäume die zugehörigen syntaktischen Strukturen.

9.1 Analysestrategien

Man kann auf viele unterschiedliche Weisen versuchen, zu gegebener Eingabe $\tilde{w} \in \Sigma^*$ einen Analysebaum bzgl. einer kontextfreien Grammatik G zu finden. Die üblichen Verfahren kann man danach klassifizieren, welche Information bzw. Teilstruktur des Baums sie benötigen, um ein weiteres Stück der Struktur zu bestimmen. Dazu ist folgende Terminologie nützlich:



[7]Immerhin sind durch $L \subseteq \Sigma^*$ die Distributionsklassen bestimmt, und diese kann man als syntaktische Kategorien auffassen. Da es aber i.a. unendlich viele Klassen von $\equiv_L$ gibt, läßt sich darauf keine *endliche* Sprachbeschreibung aufbauen.

Dabei heißen α ein REDUZIERTER und u ein TERMINALER LINKSKONTEXT sowie β ein REDUZIERTER und w ein TERMINALER RECHTSKONTEXT der Regelanwendung.

Die gebräuchlichen Analysestrategien kann man dann so klassifizieren: bevor die Fortsetzung der Teilanalyse durch die gezeigte Anwendung der Regel $A \to A_1 \dots A_n$ versucht wird, muß

(i) bei der LINKS-RECHTS-ANALYSE die Linksstruktur,

(ii) bei der RECHT-LINKS-ANALYSE die Rechtsstruktur,

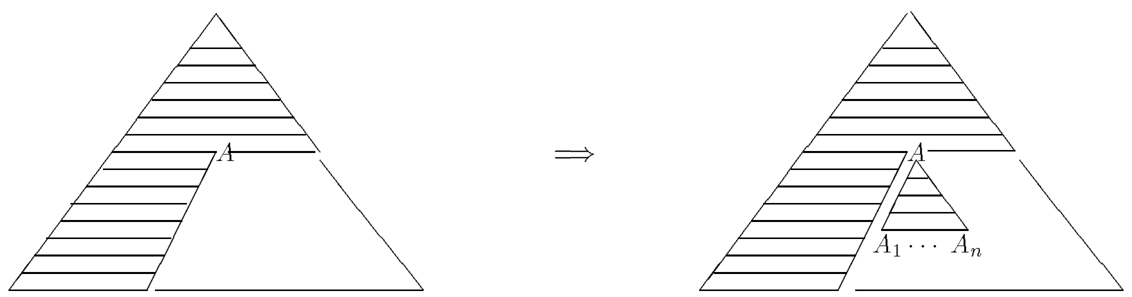
(iii) bei der TOP-DOWN- oder HYPOTHESENGESTEUERTEN ANALYSE die Vorstruktur,

(iv) bei der BOTTUM-UP- oder DATENGESTEUERTEN ANALYSE die Nachstruktur

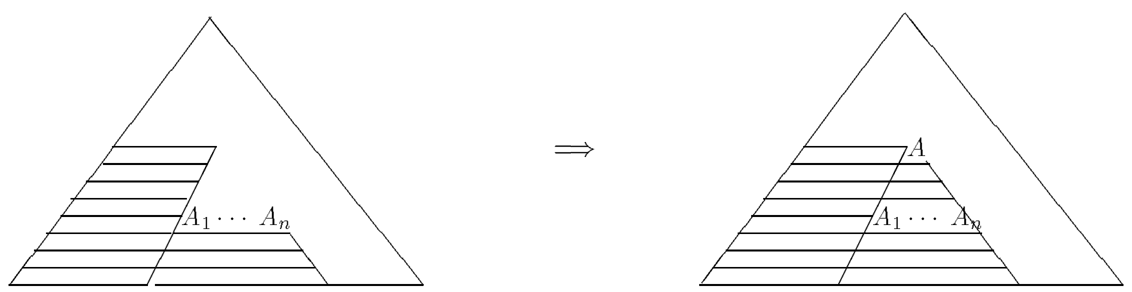
bekannt sein. Die üblichen Verfahren arbeiten von links nach rechts, sowohl bei Programmiersprachen wie bei natürlichen Sprachen. Die effizientesten Verfahren bei der Übersetzung von Programmiersprachen sind zusätzlich Bottom-Up-Verfahren, bei natürlichen Sprachen verwendet man öfter Top-Down-Verfahren.

Der Analysebaum wird in diesen Fällen wie folgt aufgebaut:

i) Left-Right Top-Down:



ii) Left-Right Bottom-Up:



1

1

Es gibt aber auch Verfahren, bei denen der Analysebaum noch anders aufgebaut wird, z.B. die sogenannte Left-Corner-Analyse.

Bei DETERMINISTISCHEN Analyseverfahren wird in Abhängigkeit von der erkannten Teilstruktur und der Eingabe eindeutig entschieden, ob die Analyse fortgesetzt werden kann, und wenn ja, durch welche Regelanwendung. Einmal erzeugte Struktur kann nicht wieder abgebaut werden.

Bei NICHT-DETERMINISTISCHEN Analyseverfahren wird unter verschiedenen Fortsetzungsmöglichkeiten "durch Ausprobieren und Rücksetzen" (Backtracking) entschieden, welche zu einer Vervollständigung der Teilanalysen führt.

Bei TABELLENGESTEUERTEN Verfahren (Chart-Parser) werden Zwischenergebnisse in Tabellen abgelegt, um Wiederholungen derselben Versuche zu ersparen.

Notation der Kellerautomaten

Wir geben die Top-Down- und Bottom-Up-Analyse als Kellerautomaten mit einer Ausgabe der syntaktischen Struktur an. Lange Befehlsfolgen der Kellerautomaten lassen sich nach der folgenden Beobachtung etwas kürzer schreiben.

Jede der Basisoperationen $push_A$, pop_A und $read_a$ einer Kellermaschine $\mathcal{M}$ nach 7.5 verändert nur eine der beiden Komponenten einer Konfiguration $(\gamma, w) \in \Gamma^* \times \Sigma^*$, den Speicherinhalt γ oder die Eingabe w. Daher spalten wir auch die Operationen in Komponenten auf,

$$push_A \simeq (A, \varepsilon), \quad pop_A \simeq (A^{-1}, \varepsilon) \quad read_a \simeq (\varepsilon, a^{-1}), \qquad A \in \Gamma,\ a \in \Sigma,$$

wobei die Anwendung auf die Konfiguration (γ, w) die komponentenweise Multiplikation im Sinne der Monoide Γ^* und Σ^* wird. Dabei sei A^{-1} eine LINKSinverse von $A \in \Gamma$ und a^{-1} eine Linksinverse von $a \in \Sigma$ ist —aber $A \cdot A^{-1}$ und $a \cdot a^{-1}$ können nicht zu ε gekürzt werden— :

$$\begin{array}{lcl} (A, \varepsilon) \cdot (\gamma, w) & = & (A\gamma, w), \\ (A^{-1}, \varepsilon) \cdot (A, \gamma) & = & (A^{-1}A\gamma, w) = (\gamma, w), \\ (\varepsilon, a^{-1}) \cdot (\gamma, aw) & = & (\gamma, a^{-1}aw) = (\gamma, w). \end{array}$$

Da sich Speicher und Eingabeband nicht überlappen, kann eine Folge von Operationen teilweise PARALLEL ausgeführt werden, getrennt in Speicherveränderungen und Lesen der Eingabe. Durch komponentenweises Zusammenfassen erhält man einen Makrobefehl, z.B.[8]

$$pop_A; read_a; push_B = (A^{-1}, \varepsilon); (\varepsilon, a^{-1}); (B, \varepsilon) = (BA^{-1}, a^{-1}).$$

Das ergibt etwas kürzere Schreibweisen für Befehlsfolgen.

[8] Da wir am linken Ende der Eingabe und des Stapels operieren, ist kehrt sich die Reihenfolge einer Befehlssequenz um, wenn man die Anwendung als Multiplikation von links darstellt:

$$(A^{-1}, \varepsilon); (\varepsilon, a^{-1}); (B, \varepsilon) = (B, \varepsilon) \cdot (\varepsilon, a^{-1}) \cdot (A^{-1}, \varepsilon) = (B \cdot \varepsilon \cdot A^{-1}, \varepsilon \cdot a^{-1} \cdot \varepsilon) = (BA^{-1}, a^{-1}).$$

9.2 Top-Down-Analyseautomaten

Die nicht-deterministische Top-Down-Analyse mit einer kontextfreien Grammatik G läßt sich leicht durch einen Kellerautomaten beschreiben. Einer kontextfreien Regel wie $A \to a\alpha$ ordnen wir eine Maschinenoperation $(\alpha A^{-1},\ a^{-1})$ zu, die über

$$\kappa \underset{(\alpha A^{-1}\, a^{-1})}{\vdash} \kappa' \iff \text{Es gibt } (\beta, v) \in \Gamma^* \times \Sigma^* \text{ mit } \kappa = (A\beta, av) \text{ und } \kappa' = (\alpha\beta, v)$$

eine Relation $\underset{(\alpha A^{-1}, a^{-1})}{\vdash}$ zwischen Konfigurationen $\kappa, \kappa' \in (N \cup \Sigma)^* \times \Sigma^*$ liefert.

Definition 9.1 Der Top-Down-Analyseautomat zur kontextfreien Grammatik $G = (N, \Sigma, P, S)$ ist der Kellerautomat $\mathcal{M}^G := (N \cup \Sigma, \Sigma, M, S)$ mit $M := (\mathit{predict} + \mathit{check})^*$, wobei

$$\mathit{predict} := \{\, (\alpha A^{-1}, \varepsilon) \mid A \to \alpha \in P \,\}, \quad \mathit{check} := \{\, (a^{-1}, a^{-1}) \mid a \in \Sigma \,\}.$$

Die hierbei verwendeten Operationen lassen sich natürlich aus den Basisoperationen zusammensetzen, z.B. $(a^{-1}, a^{-1}) = \mathit{pop}_a \cdot \mathit{read}_a$.

Satz 9.2 Sei $\mathcal{M}^G$ der Top-Down-Analyseautomat für G. Dann ist $L(G) = L(\mathcal{M}^G)$.

Beweis: $L(G) \subseteq L(M^G)$: Seien $w \in \Sigma^*$ und $\alpha \in N(N \cup \Sigma)^* \cup \{\varepsilon\}$ mit $S \overset{*}{\underset{G}{\Longrightarrow}} w\alpha$. Wir zeigen $(S, w) \underset{M}{\vdash} (\alpha, \varepsilon)$ durch Induktion über die Länge der Ableitung $S \overset{n}{\underset{G}{\Longrightarrow}} w\alpha$, woraus die Behauptung mit $\alpha = \varepsilon$ folgt.

$n = 0$: Dann ist $w = \varepsilon, \alpha = S$, und da M die Form K^* hat, gilt $(S, w) \underset{M}{\vdash} (S, w)$.

$n > 0$: Sei $S \overset{n}{\underset{G}{\Longrightarrow}} uA\beta \overset{1}{\underset{G}{\Longrightarrow}} u\alpha\beta$ mit $u \in \Sigma^*$. Nach Induktion gilt $(S, u) \underset{M}{\vdash} (A\beta, \varepsilon)$. Wegen $(A \to \alpha) \in P$ gilt weiter $(A\beta, \varepsilon) \underset{\mathit{predict}}{\vdash} (\alpha\beta, \varepsilon)$, insgesamt also $(S, u) \underset{M}{\vdash} (\alpha\beta, \varepsilon)$. Ist nun $w\gamma = u\alpha\beta = uv\gamma$ mit $\gamma \in N(N \cup \Sigma)^* \cup \{\varepsilon\}$, und $v \in \Sigma^*$, so folgt $(S, w) = (S, uv) \underset{M}{\vdash} (\alpha\beta, v) = (v\gamma, v) \underset{\mathit{check}}{\vdash}^{*} (\gamma, \varepsilon)$.

$L(\mathcal{M}^G) \subseteq L(G)$: Da $M = (\mathit{predict} + \mathit{check})^*$, haben wir $(S, w) \underset{M}{\vdash} (\alpha, \varepsilon)$ genau dann, wenn $(S, w) \underset{K}{\vdash}^{n} (\alpha, \varepsilon)$ für ein $n \geq 0$, wobei $K = (\mathit{predict} + \mathit{check})$. Jede solche Operationsfolge zerfällt in *predict*- und *check*-Stücke. Daher zeigt man für jedes $\beta \in N(N \cup \Sigma)^* \cup \{\varepsilon\}$:

(i) Wenn $(A, w) \underset{\mathit{predict}}{\vdash}^{*} (v\beta, u)$, so ist $A \overset{*}{\underset{G}{\Longrightarrow}} v\beta$ und $u = w$.

(ii) Wenn $(v\beta, u) \underset{\mathit{check}}{\vdash}^{*} (\gamma, w)$, so gibt es $v_1, v_2 \in \Sigma^*$ mit

$$(v\beta, u) = (v_1 v_2 \beta, v_1 w) = (v_1 \gamma, v_1 w).$$

Man überlegt sich leicht, daß die Behauptung daraus folgt. □

Beispiel 9.3 (Fortsetzung von 8.3) Betrachte den Top-Down-Automat $\mathcal{M}^G$ zur Grammatik

$$G = (\{S\}, \{a, b\}, \{S \to \varepsilon, S \to aSb\}, S).$$

Wir erhalten $M = (\mathit{predict} + \mathit{check})^*$ mit

$$\mathit{predict} = \{(S^{-1}, \varepsilon), (aSbS^{-1}, \varepsilon)\}, \qquad \mathit{check} = \{(a^{-1}, a^{-1}), (b^{-1}, b^{-1})\}.$$

Bei der Ausgangkonfiguration $(S, aabb)$ kann $\mathcal{M}^G$ höchstens einen *predict*-Schritt machen, und man erhält je nach Wahl der Operation

$$(S, aabb) \vdash_{(S^{-1},\varepsilon)} (\varepsilon, aabb), \quad \text{oder} \quad (S, aabb) \vdash_{(aSbS^{-1},\varepsilon)} (aSb, aabb).$$

Der linke Fall ist eine Sackgasse. Im rechten Fall kann man mit $\vdash_{\mathit{check}}$ zu (Sb, abb) übergehen, und so weiter:

$$\begin{array}{llll} (Sb, abb) & \vdash_{\mathit{predict}} (aSbb, abb) & \vdash_{\mathit{check}} (Sbb, bb) & \\ & \vdash_{\mathit{predict}} (\ bb, bb) & \vdash_{\mathit{check}} (b, b) & \vdash_{\mathit{check}} (\varepsilon, \varepsilon) \end{array}$$

Man sieht, wie das *nicht-deterministische* Raten des Berechnungswegs bestimmt, ob man eine Analyse findet.

Wir würden gerne das blinde Raten einer Produktion etwas sorgfältiger von der Eingabe abhängig machen: *falls* das nächste Eingabesymbol a ist, sollte man unter $S \to \varepsilon$ und $S \to aSb$ stets die zweite Regel wählen.

Dazu modifizieren wird den Automaten, indem wir eine VORAUSSCHAU einbauen:

$$\begin{array}{lll} \mathit{lookahead} & := & \{(\varepsilon, aa^{-1}), (\varepsilon, bb^{-1})\} \\ \mathit{predict\text{-}with\text{-}lookahead} & := & \{(aSbS^{-1}, \varepsilon) \circ (\varepsilon, aa^{-1}), (S^{-1}, \varepsilon) \circ (\varepsilon, bb^{-1})\} \\ & = & \{(aSbS^{-1}, aa^{-1}), (S^{-1}, bb^{-1})\} \end{array}$$

Zwar machen wir hierbei insgesamt keine echte Veränderung der Eingabe, aber das Schreiben auf der Eingabe hatten wir nicht als Operation zugelassen. Deshalb kombinieren wir die Schritte mit den einzig möglichen Leseschritten, die folgen müssen:

$$\begin{array}{lll} \mathit{predict}' & = & \{(a^{-1}, a^{-1}) \circ (aSbS^{-1}, aa^{-1}), (b^{-1}, b^{-1}) \circ (S^{-1}, bb^{-1})\} \\ & = & \{(SbS^{-1}, a^{-1}), (b^{-1}S^{-1}, b^{-1})\} \\ \mathcal{M}' & = & (\mathit{predict}' + \mathit{check})^* \end{array}$$

Beachte, daß die Operation $(b^{-1}S^{-1}, b^{-1})$ zulässig ist, da sie über $(b^{-1}, \varepsilon) \circ (S^{-1}, b^{-1})$ aus den Basisoperationen zusammengesetzt werden kann.

Damit haben wir erreicht, was wir wollten: $\mathcal{M}'$ arbeitet *deterministisch*. Die Berechnung $(S, a^n b^n) \vdash_{\overline{M'}} (\varepsilon, \varepsilon)$ verläuft wie folgt:

$$\begin{array}{lll} (S, a^n b^n) & \vdash_{\overline{(SbS^{-1}, a^{-1})}}{}^{*} & (Sb, a^{n-1}b^n) \\ & \vdash_{\overline{(SbS^{-1}, a^{-1})}}{}^{*} & (Sb^n, b^n) \\ & \vdash_{\overline{(b^{-1}S^{-1}, b^{-1})}} & (b^{n-1}, b^{n-1}) \\ & \vdash_{\overline{check}}{}^{*} & (\varepsilon, \varepsilon) \end{array}$$

Es ist aber nicht für jede kontextfreie Sprache möglich, einen erkennenden Kellerautomaten anzugeben, der deterministisch arbeitet.

9.3 Bottom-Up-Analyseautomaten

Für die nicht-deterministische Bottom-Up-Analyse können wir ebenfalls einen einfachen Kellerautomaten angeben. Es ist hier aber anschaulicher, wenn man am *rechten* Ende des Kellers operiert, d.h. wir benutzen die Grundbefehle wie in

$$(\alpha, w) \vdash_{\overline{push_A}} (\alpha A, w) \vdash_{\overline{pop_A}} (\alpha, w) \quad \text{und} \quad (\alpha, aw) \vdash_{\overline{read_a}} (\alpha, w).$$

Definition 9.4 Der Bottom-Up-Analyseautomat zur kontextfreien Grammatik $G = (N, \Sigma, P, S)$ ist der Kellerautomat $M_G = (N \cup \Sigma, \Sigma, M, \varepsilon)$ mit

$$\begin{array}{lcl} M & := & (\mathit{shift} + \mathit{reduce})^* \cdot pop_S \\ \mathit{shift} & := & \{(a, a^{-1}) \mid a \in \Sigma\} \\ \mathit{reduce} & := & \{\, (\alpha^{-1}A, \varepsilon) \mid (A \to \alpha) \in P \,\} \end{array}$$

Hierbei ist $(\alpha^{-1}A, \varepsilon)$ für $(A \to \alpha) = (A \to A_1 \cdots A_n)$ durch $pop_{A_n} \cdot pop_{A_{n-1}} \cdots pop_{A_1} \cdot push_A$ mit den Basisbefehlen darstellbar.

Satz 9.5 Sei $\mathcal{M}_G$ der Bottom-Up-Analyseautomat zur Grammatik G. Dann ist $L(\mathcal{M}_G) = L(G)$.

Beweis: $L(\mathcal{M}_G) \subseteq L(G)$: Durch Induktion über die Anzahl der Iterationsschritte sieht man

(i) $(\alpha, w) \vdash_{\overline{shift}}{}^{*} (\beta, v) \quad \Longrightarrow \quad \exists u \in \Sigma^* \, (w = uv \;\wedge\; \beta = \alpha u),$ und

(ii) $(\beta, v) \vdash_{reduce}^{*} (\alpha, w) \quad \Longrightarrow \quad v = w \;\wedge\; \alpha \overset{*}{\Longrightarrow}_G \beta.$

Sei $(\varepsilon, w) \vdash_M (\varepsilon, \varepsilon)$. Dann gibt es wegen $(\mathit{shift}+\mathit{reduce})^* = (\mathit{shift}^* \cdot \mathit{reduce}^*)^*$ eine Berechung der Form

$$\begin{array}{rll}
(\varepsilon, w) = (\alpha_0, u_0 v_0) & \vdash_{shift}^{*} & (\alpha_0 u_0, v_0) \\
& \vdash_{reduce}^{*} & (\alpha_1, v_0) = (\alpha_1, u_1 v_1) \\
& \vdash_{shift}^{*} & (\alpha_1 u_1, v_1) \\
& \vdots & \\
& \vdash_{reduce}^{*} & (\alpha_k, v_{k-1}) = (S, \varepsilon) \\
& \vdash_{pop_S} & (\varepsilon, \varepsilon),
\end{array}$$

worin nach i) $\alpha_{m+1} \overset{*}{\Longrightarrow}_G \alpha_m u_m$ für jedes m gilt. Also ist

$$S = \alpha_k \overset{*}{\Longrightarrow}_G \alpha_{k-1} u_{k-1} \overset{*}{\Longrightarrow}_G \alpha_{k-2} u_{k-2} u_{k-1} \overset{*}{\Longrightarrow}_G \ldots \overset{*}{\Longrightarrow}_G \alpha_0 u_0 u_1 \ldots u_{k-1}.$$

Wegen $\alpha_0 = \varepsilon = v_{k-1}$ und $w = u_0 v_0 = u_0 u_1 v_1 = \ldots = u_0 u_1 \ldots u_{k-1} v_{k-1}$ ist damit $w \in L(G)$ gezeigt.

$L(G) \subseteq L(M_G)$: Zu $w \in L(G)$ gibt es eine Rechtsableitung

$$S \Longrightarrow_G \alpha_1 A_1 u_1 \Longrightarrow_G \alpha_2 A_2 u_2 u_1 \Longrightarrow_G \ldots \Longrightarrow_G \alpha_{k-1} A_{k-1} u_{k-1} \ldots u_2 u_1 \Longrightarrow_G u_k u_{k-1} \ldots u_2 u_1,$$

in der im j-ten Schritt das am weitesten rechts stehende Nonterminal A_j der Satzform $\alpha_j A_j u_j \ldots u_1$ ersetzt wird, also $\alpha_j \in (N \cup \Sigma)^*$ und $u_j \in \Sigma^*$. Durch Induktion über die Länge dieser Ableitung zeigt man, daß

$$\begin{array}{rll}
(\varepsilon, w) = (\varepsilon, u_k \ldots u_1) & \vdash_{shift}^{*} & (u_k, u_{k-1} \ldots u_1) \\
& \vdash_{reduce}^{*} & (\alpha_{k-1} A_{k-1}, u_{k-2} \ldots u_1) \\
& \vdash_{shift}^{*} & (\alpha_{k-1} A_{k-1} u_{k-2}, u_{k-3} \ldots u_1) \\
& \vdots & \\
& \vdash_{reduce}^{*} & (\alpha_1 A_1, u_1) \\
& \vdash_{shift}^{*} & (\alpha_1 A_1 u_1, \varepsilon) \\
& \vdash_{reduce}^{*} & (S, \varepsilon) \\
& \vdash_{pop_S} & (\varepsilon, \varepsilon),
\end{array}$$

so daß $(\varepsilon, w) \vdash_M (\varepsilon, \varepsilon)$ gilt. □

Beispiel 9.6 (Fortsetzung von 8.3) Sei G wieder durch $S \to \varepsilon, S \to aSb$ gegeben. Wir erhalten

$$\mathit{reduce} = \{(S, \varepsilon), ((aSb)^{-1} S, \varepsilon)\}.$$

Für die Eingabe $aabb$ berechnen wir:

$$
\begin{array}{lll}
(\varepsilon, aabb) & \vdash_{\overline{shift}} & (a, abb) \\
& \vdash_{\overline{shift}} & (aa, bb) \\
& \vdash_{\overline{reduce}} & (aaS, bb) \\
& \vdash_{\overline{shift}} & (aaSb, b) \\
& \vdash_{\overline{reduce}} & (aS, b) \\
& \vdash_{\overline{shift}} & (aSb, \varepsilon) \\
& \vdash_{\overline{reduce}} & (S, \varepsilon), \\
& \vdash_{\overline{pop_S}} & (\varepsilon, \varepsilon).
\end{array}
$$

Also ist $(\varepsilon, aabb) \vdash_{\overline{M_G}} (S\ \varepsilon)$ und damit $aabb \in L(\mathcal{M}_G)$.

9.4 Aufbau der syntaktischen Strukturen

Es wurde bisher nicht diskutiert, wie man durch die Analyseautomaten M^G und M_G und zu vorgegebener Eingabe w einen Syntaxanalysebaum oder G-Ableitungsbaum für w konstruieren kann.

Wir können annehmen, daß die Regeln der Grammatik numeriert sind. Wir geben nun während der Berechnung die Regelnummern, die von den verwendeten *predict*- bzw. *reduce*-Schritten entsprechen, auf einem Ausgabeband aus.

Dazu ändern wir die Grundoperationen wie folgt:

$$
\begin{array}{lll}
\textit{predict-mit-Ausgabe} & := & \{\, (\alpha A^{-1}, \varepsilon, i) \mid i : (A \to \alpha) \in P \,\} \\
\textit{reduce-mit-Ausgabe} & := & \{\, (\alpha^{-1} A, \varepsilon, i) \mid i : (A \to \alpha) \in P \,\}.
\end{array}
$$

Wird $w \in L(G)$ vom Automaten mit Ausgabe erkannt, so beschreibt die Folge der ausgegebenen Regelnummern bei M^G eine Linksableitung, bei M_G eine Rechtsableitung. Diese entsprechen in eindeutiger Weise einem G-Ableitungsbaum von w. Die akzeptierende Berechnung konstruiert also einen Analysebaum von w bezüglich G.

9.5 Tabellenanalyse nach Cocke, Younger, Kasami

Die Erkennung und Analyse kontextfreier Sprachen ist besonders einfach für Grammatiken in Chomsky-Normalform, wenn alle Regeln die Form $A \to B{\cdot}C$ oder $A \to a$ mit $A, B, C \in N$ und $a \in \Sigma$ haben. In diesem Fall wissen wir

(i) alle zusammengesetzten Ausdrücke w der Kategorie A bestehen aus *genau zwei* unmittelbaren Teilausdrücken w_1 und w_2 von Arten B und C, und

(ii) alle unzusammengesetzten Ausdrücke w der Kategorie A sind Buchstaben $a \in \Sigma$.

Daraus folgt, daß die Erkennung und Analyse in einfacher Weise durch Rekursion über die Wortlänge betrieben werden kann: die Tiefe und Breite[9] eines Analysebaums sind durch die Wortlänge beschränkt. Beachte, daß dies bei kontextfreien Grammatiken mit Ketten- und Löschregeln nicht der Fall ist.

Genauer gesagt, erfolgt die Analyse nicht rekursiv über die Wortlänge, sondern rekursiv über die Teilwortrelation

$$v \leq w :\Longleftrightarrow \ \exists u_1, u_2 \in \Sigma^* \, w = u_1 v u_2.$$

Die grammatischen Eigenschaften von $w \in \Sigma^*$, d.h. die Kategorien $A \in N$ mit $w \in L(A)$, sind bei einer kontextfreien Grammatik nämlich durch die grammatischen Eigenschaften aller (echten) Teilwörter v von w eindeutig bestimmt. Die Rekursion über die Teilwortrelation hat den Vorteil, daß es zu einem Wort $w \in \Sigma^*$ der Länge n zwar $O(|\Sigma|^{n-1})$ viele kürzere Wörter gibt, aber nur $|\{\,(i,j) \mid 0 \leq i < j \leq n\,\}| = O(n^2)$ viele Teilwörter.

Da verschiedene (auch bzgl. $\leq$ unvergleichbare) Teilwörter gewisse Teilwörter gemeinsam haben können, kann man außerdem eine Wiederholung von schon geleisteter Analysearbeit vermeiden, wenn man sich eine Tabelle von Zwischenergebnissen anlegt.

Der Algorithmus von Cocke, Younger und Kasami ist ein Erkennungsverfahren, das man aber leicht zu einem Analyseverfahren ausbauen kann, indem man mehr Information in der Hilfstabelle speichert.

Satz 9.7 Sei G eine kontextfreie Grammatik in Chomsky-Normalform, $w := a_1 \cdots a_n \in \Sigma^*$, mit $a_i \in \Sigma$. Konstruiere eine Tabelle $M = \{\, M(i,j) \mid 0 \leq i < j \leq n \,\}$ wie folgt:

for $j = 1$ to n do
  $M(j-1, j) := \{\, A \in N \mid A \to a_j \in P \,\}$;
  if $j > 1$ then
    for $i = j-2$ to 0 do
      $M(i,j) := \bigcup_{i<k<j} \{\, A \in N \mid A \to BC \in P, B \in M(i,k), C \in M(k,j) \,\}$
      od
  od

Dann gilt für $0 \leq i < j \leq n$:

$$A \in M(i,j) \iff a_{i+1} \cdots a_j \in L(A).$$

Insbesondere ist daher $w \in L(G)$ genau dann der Fall, wenn $S \in M(0,n)$ ist.

[9] d.h. maximale Anzahl von Knoten auf einem Querschnitt durch alle Äste

Beweis: Wir zeigen die Behauptung durch Induktion über $j - (i+1)$.

Beh 1: $A \in M(i,j) \implies a_{i+1} \cdots a_j \in L(A)$.

Fall $i+1 = j$: Zu $A \in M(i,j) = M(j-1,j)$ ist $A \to a_j \in P$, also $a_j \in L(A)$.

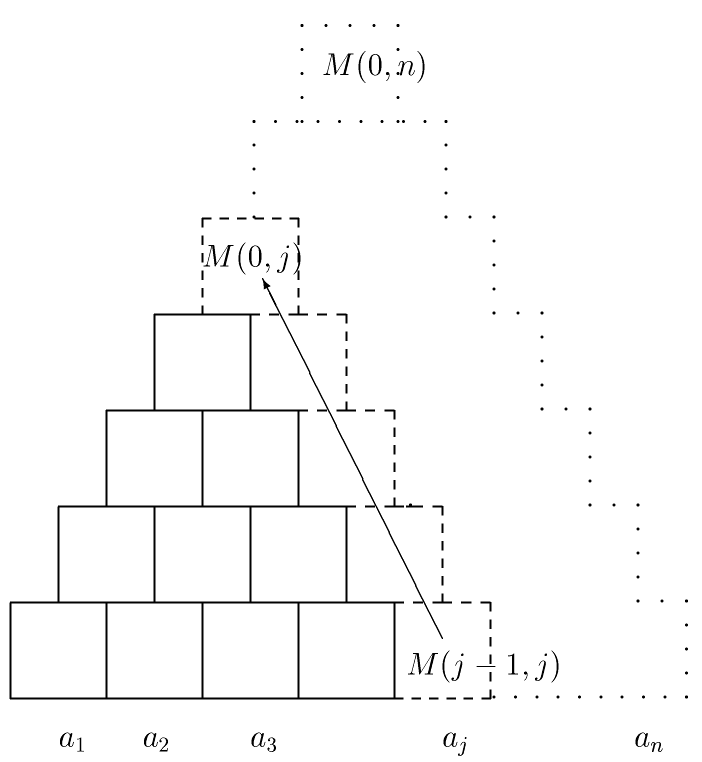
Fall $i+1 < j$: Zu $A \in M(i,j)$ gibt es $i < k < j$ und $A \to BC \in P$ mit $B \in M(i,k)$ und $C \in M(k,j)$. Nach Induktion ist $a_{i+1} \cdots a_k \in L(B)$ und $a_{k+1} \cdots a_j \in L(C)$. Daher ist $a_{i+1} \cdots a_j \in L(B) \cdot L(C) \subseteq L(A)$.

Beh. 2: $a_{i+1} \cdots a_j \in L(A) \implies A \in M(i,j)$.

$i+1 = j$: Da G in Chomsky-Normalform ist, gibt es keine löschbaren Nonterminale, so daß in einer Ableitung $A \overset{*}{\Longrightarrow}_G a_j$ keine verzweigende Regel vorkommt. Da G auch keine Kettenregeln $B \to C$ hat, kann die Ableitung nur aus der Anwendung einer lexikalischen Regel $A \to a_j$ bestehen. Nach Konstruktion ist dann $A \in M(j-1,j) = M(i,j)$.

$i+1 < j$: Sei $a_{i+1} \cdots a_j \in L(A)$. Da $j > 1$ ist und die Gleichung für A nur Summanden der Form $B \cdot C$ oder a mit Variablen B, C oder Konstanten $a \in \Sigma$ hat, muß es einen Summanden BC mit $a_{i+1} \cdots a_j \in L(B) \cdot L(C) \subseteq L(A)$ geben, also $a_{i+1} \cdots a_j = u \cdot v$ für geeignete $u \in L(B)$ und $v \in L(C)$. Da aber $\varepsilon \notin L(X)$ für alle Variablen X ist, ist dies eine Zerlegung von $a_{i+1} \cdots a_j$ in ein echtes Präfix und Suffix, so daß es ein k mit $i < k < j$ und $u = a_{i+1} \cdots a_k$ und $v = a_{k+1} \cdots a_j$ gibt. Nach Induktion ist $B \in M(i,k)$ und $C \in M(k,j)$, und aus der Konstruktion von $M(i,j)$ folgt $A \in M(i,j)$. □

Die hier beschriebene Version ist ein "on-line"-Verfahren, bei dem nach Lesen eines Präfixes $a_1 \cdots a_{j-1}$ alle Felder $M(i,k)$ mit $0 \le i < k \le j-1$ aufgebaut werden. Nach Lesen von a_j werden dann die Felder $M(j-1,j), M(j-2,j), \ldots, M(0,j)$ bestimmt:



Bemerkung 9.8 Diese Methode der Erkennung ist nicht nur für kontextfreie Sprachen möglich, sondern für alle durch Rekursion über die Teilwortrelation definierbaren Sprachen. Es kommt nur darauf an, daß die grammatischen Eigenschaften eines Wortes w, d.h. die endlich vielen, durch die Grammatik definierten Kategorien (als Prädikate über Zeichenreihen verstanden), berechenbar von den grammatischen Eigenschaften der echten Teilwörter von w abhängen.

Beispielsweise kann man auch für $L = L(G_1) \cap L(G_2)$ mit kontextfreien Grammatiken G_1 und G_2 ein Erkennungsverfahren wie das von Cocke u.a. angeben, bei dem in $M(i,j)$ die auf $a_{i+1} \cdots a_j$ zutreffenden Kategorien von G_1 und G_2 sammelt (für $N_1 \cap N_2 = \emptyset$). Im allgemeinen ist eine solche Sprache L nicht kontextfrei.

Die Forderung von Satz 9.7, die Grammatik G solle in Chomsky-Normalform sein, ist nur für die Komplexität des Verfahrens von Bedeutung. Sie sorgt dafür, daß $M(i,j)$ in $O(|j-i| \cdot |P| \cdot |N|)$ und damit M in $O(|w|^3 \cdot |G|^2)$ Schritten berechnet werden kann.

9.6 Tabellenanalyse nach Earley

Top-Down-Verfahren der Syntaxanalyse haben den Nachteil, daß auch unter Verwendung einer Vorausschau die Analyse nicht terminiert, wenn die Sprachdefinition eine Linksrekursion enthält. Bottom-Up-Verfahren haben im allgemeinen den Nachteil, daß sie auch viele Analysen erzeugen, an denen man — in der jeweiligen Situation — nicht interessiert ist. Dies kann man in gewissem Maße aber verhindern, indem man situationsabhängig vorgibt, welche Kategorie man für den folgenden Ausdruck "erwartet".

Verfahren, die die Vorteile der durch die Eingabedaten gesteuerten Bottom-Up-Methode mit denen der durch die Erwartung oder Hypothesen gesteuerten Top-Down-Methode verbinden und die jeweiligen Nachteile teilweise vermeidet, beruhen auf Tabellen von Zwischenergebnissen und werden auch Tabellenverfahren genannt (engl. chart parser). Dazu gehört das Verfahren von J. Earley[6], das wir ohne Korrektheitsbeweis vorstellen.

Sei $G = (N, \Sigma, P, S)$ eine kontextfreie Grammatik, und $w = a_1 \cdots a_n \in \Sigma^*$ eine zu analysierende Eingabe. Das Verfahren von J. Earley erzeugt Listen $\mathit{Items}_1, \ldots, \mathit{Items}_n$ von Einträgen, so daß folgende Bedingungen äquivalent sind:

(i) $[A \to \alpha \cdot \beta, i, v] \in \mathit{Items}_j$

(ii) (a) $(A \to \alpha\beta) \in P$,

(b) $0 \leq i \leq j \leq n$,

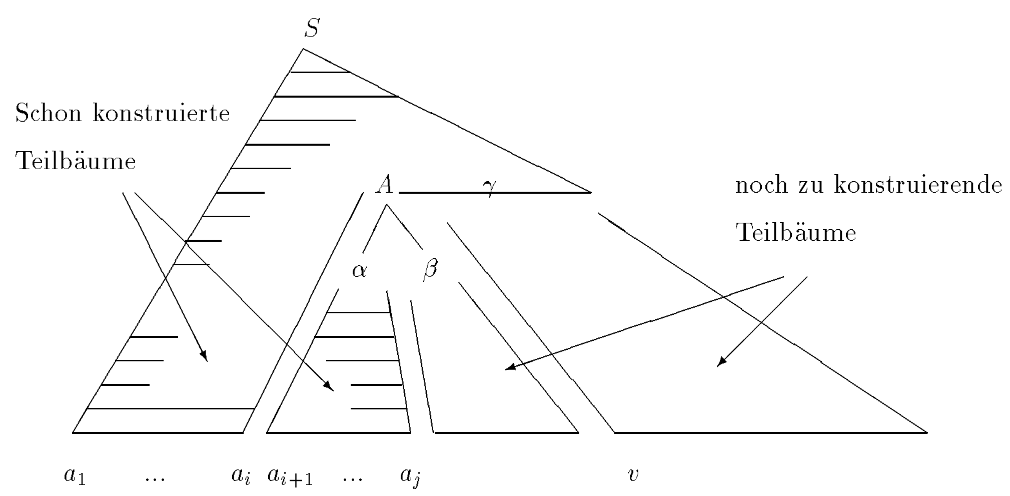
(c) Es gibt $\gamma \in (N \cup \Sigma)^*$ mit $v \in \mathit{first}_k(\gamma)$ und eine Linksableitung

$$S \overset{*}{\underset{G}{\Longrightarrow}} a_1 \ldots a_i A \gamma \underset{G}{\Longrightarrow} a_1 \ldots a_i \alpha\beta\gamma \overset{*}{\underset{G}{\Longrightarrow}} a_1 \ldots a_i a_{i+1} \ldots a_j \beta\gamma.$$

Eine LINKSABLEITUNG ist dabei eine Ableitung, in der in jedem Schritt das am weitesten links stehende Nonterminal der Satzform durch eine rechte Regelseite ersetzt wird, und

$$first_k(\gamma) := \{w \in \Sigma^* \mid \gamma \overset{*}{\underset{G}{\Longrightarrow}} w,\ |w| \leq k\}.$$

Die Bedingung wird durch folgendes Bild anschaulich klar:



Man verwendet v zum Vergleich mit der Vorausschau um k noch nicht gelesene Buchstaben der Resteingabe (im Fall $\beta = \varepsilon$). Ob die Eingabe w zu $L(G)$ gehört, erkennt man daran, ob ein Eintrag der Form

$$[S \to \alpha\cdot, 0, \varepsilon] \in Items_n$$

vorkommt. Falls dies nicht der Fall ist — insbesondere, wenn eine der Itemmengen leer ist, — gehört w nicht zu $L(G)$. Im anderen Fall gehört w zu $L(G)$, und man kann aus der Tabelle eine Analyse von w rekonstruieren.

Definition 9.9 (Earley 1971) Die Menge der zur Grammatik G und Eingabe w aufzustellenden Itemmengen ist die kleinste Familie, die folgenden Abschlußbedingungen genügt:

(i) Predictor (Erzeuge den Anfang möglicher Analysen von $a_{j+1} \cdots a_n$):

$$\left.\begin{array}{l}[A \to \alpha \cdot B\gamma, i, u] \in Items_j\\ (B \to \beta) \in P,\\ v \in first_k(\gamma u)\end{array}\right\} \quad\Rightarrow\quad [B \to \cdot\beta, j, v] \in Items_j$$

(ii) Completer (Vervollständige einen partiellen Baum): ($\gamma = \varepsilon \Rightarrow v = a_{j+1} \ldots a_h$)

$$\left.\begin{array}{l}[B \to \beta\cdot, k, u] \in Items_j,\\ [A \to \alpha \cdot B\gamma, i, v] \in Items_k\end{array}\right\} \quad\Rightarrow\quad [A \to \alpha B \cdot \gamma, i, v] \in Items_j$$

(iii) Scanner (Lies den nächsten Buchstaben der Eingabe):

$$[A \to \alpha \cdot a\beta, i, v] \in \mathit{Items}_{j-1},\ a = a_j \qquad \Rightarrow \qquad [A \to \alpha a \cdot \beta, i, v] \in \mathit{Items}_j.$$

Zur Initialisierung werden alle $\mathit{Items}_j := \emptyset$ gesetzt, bis auf

$$\mathit{Items}_0 := \{[\tilde{S} \to \cdot S\$, 0, \$^k]\}$$

mit einem in G nicht vorkommenden Endsymbol \$ und einer Pseudoregel $\tilde{S} \to S\$$, wobei S das Startsymbol von G ist.

Den Aufbau der Itemmengen kann man durch eine geschachtelte Schleife verwirklichen:

Algorithmus 9.10 Algorithmus von Earley (ohne Vorausschau)

EINGABE: Eine kontextfreie Grammatik G und ein Wort $w = a_1 \ldots a_n \in \Sigma^*$.

AUSGABE: Die Liste der Itemmengen $\mathit{Items}_0, \ldots, \mathit{Items}_n$ wie oben charakterisiert.

PROGRAMM:

Initialize :	$\mathit{expect}(0) := \{S\}$; $\mathit{Items}_0 := \emptyset; \ldots; \mathit{Items}_n := \emptyset$;
	for $j = 0, 1, 2, \ldots, n$ do:
predict(j) :	for $B \in \mathit{expect}(j)$ and $(B \to \beta) \in P$ do
	$\quad$ add $[B \to \cdot\beta, j]$ to Items_j;
	$\quad$ if $\beta = C\gamma$, add C to $\mathit{expect}(j)$;
	$\quad$ (until $\mathit{expect}(j)$ and Items_j do not change.)
scan(j) :	for $i = j, j-1, \ldots, 0$ do
	$\quad$ for $[A \to \alpha \cdot a_{j+1}\beta, i] \in \mathit{Items}_j$ do
	$\quad\quad$ add $[A \to \alpha a_{j+1} \cdot \beta, i]$ to Items_{j+1};
	$\quad\quad$ if $\beta = C\gamma$, add C to $\mathit{expect}(j+1)$;
complete(j) :	for $i = j, j-1, \ldots, 0$ do
	$\quad$ for $k = j, j-1, \ldots, i$ do
complete(i,k,j) :	$\quad\quad$ for $[A \to \alpha \cdot B\gamma, i] \in \mathit{Items}_k$ and $[B \to \beta\cdot, k] \in \mathit{Items}_j$ do
	$\quad\quad\quad$ add $[A \to \alpha B \cdot \gamma, i]$ to Items_j;
	$\quad\quad\quad$ if $\gamma = C\delta$, add C to $\mathit{expect}(j)$;

Die Einträge werden oft in Form einer ITEM-TABELLE $\{\mathit{Items}(i,j) \mid 0 \le i \le j \le n\}$ gespeichert, mit

$$[A \to \alpha \cdot \beta, v] \in \mathit{Items}(i,j) \iff [A \to \alpha \cdot \beta, i, v] \in \mathit{Items}_j.$$

Man kann zeigen (vgl. Aho/Ullman[1]), daß für jede Eingabe w der Länge n

(i) die Tabelle der Itemmengen

(a) in $O(n^3)$ Schritten erstellt ist, wenn G eine beliebige kontextfreie Grammatik ist,

(b) in $O(n^2)$ Schritten erstellt ist, falls G beschränkt mehrdeutig ist;

(c) in $O(n)$ Schritten erstellt ist, falls es eine nicht von n, sondern nur von G abhängige Schranke für die Anzahl der Einträge in jedem $Items_j$ gibt;

(ii) in $O(n^2)$ Schritten eine Analyse von w aus der Tabelle der Itemmengen erstellt werden kann.

Insbesondere können nach γ) auch für einige mehrdeutige Grammatiken Verfahren angegeben werden, mit denen man in $O(|w|)$ Schritten feststellen kann, ob $w \in L(G)$ (mit Vorausschau).

Es gibt natürlich kontextfreie Grammatiken G mit Wörtern $w \in L(G)$, die $O(2^{|w|})$ verschiedene Ableitungsbäume haben.

Bemerkung 9.11 Zur syntaktischen Analyse von Sätzen einer natürlichen Sprache ist weniger die Satzlänge als vielmehr die Größe der Grammatik der wesentliche Faktor. Der Aufbau der Itemmengen erfordert $O(|w|^3 \cdot |G|^2)$ viele Schritte, wenn man einen konstanten Aufwand für das Lesen und Schreiben in die Itemmengen annimmt. Mit $|G|$ ist dabei die Summe der Längen der rechten Regelseiten gemeint.

Wir beschreiben nun eine anschauliche und etwas optimierte Variante der Earley-Analyse, bei der statt der gepunkteten Regeln gleich Teile von Analysebäumen erzeugt werden.[10]

Dazu muß allerdings ein Problem beachtet werden: wenn G Kettenregeln oder Löschregeln hat (aber auch nur dann), kann es zu einem Wort unendliche viele verschiedene Analysebäume geben (mit unbeschränkt tiefen bzw. unbeschränkt breiten Bäumen). Eine Analyse, die alle Bäume erzeugt, ist daher in endlich vielen Schritten nicht möglich.

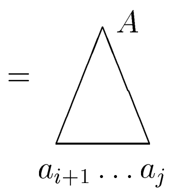
Dieses Problem können wir aber umgehen, indem wir für Löschungen $A \overset{*}{\underset{G}{\Longrightarrow}} \varepsilon$ und für Kettenableitungen $A \overset{*}{\underset{G}{\Longrightarrow}} B$ jeweils einen KANONISCHEN Ableitungsbaum (z.B. den zur kürzesten Linksableitung) auswählen. Alle anderen Möglichkeiten, A zu löschen oder zu B zu machen, werden ignoriert. Man überlege sich, daß es zu jedem $w \in L(G)$ nur endlich viele Analysebäume gibt, in denen alle Löschungen und Kettenableitungen "kanonisch" sind. Im Folgenden seien

$$\begin{array}{c} A \\ | \\ \varepsilon \end{array} \quad \text{und} \quad \begin{array}{c} A \\ | \\ B \end{array}$$

[10] Es ist möglich, die Mengen dieser Bäume so darzustellen, daß sich verschiedene Analysen isomorphe Teilbäume teilen. Das reduziert den Speicherplatz gegenüber einer vollen Darstellung aller (ggf. exponentiell vieler) Bäume erheblich.

die kanonischen Bäume zu $A \Rightarrow^* \varepsilon$ und $A \Rightarrow^* B$. Wir bauen statt der Itemmengen jetzt Mengen von Bäumen auf. Dabei unterscheiden wir zwischen VOLLSTÄNDIGEN Bäumen

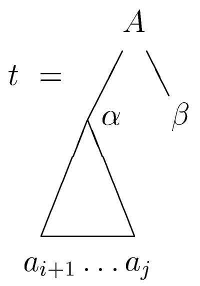
$t =$



$\in \ ctrees(i, j, A)$

und PARTIELLEN Bäumen

$t =$



$\in \ ptrees(i, j, A \to \alpha \cdot \beta).$

Zur Vereinfachung der Vervollständigungsschritte sei auch $\begin{array}{c} a \\ | \\ a \end{array}$ als kanonischer Baum zugelassen. Zur Verwendung der kanonischen Teilbäume müssen die Mengen

$$\{A \mid A \in N,\ A \overset{*}{\underset{G}{\Longrightarrow}} \varepsilon\} \quad \text{und} \quad \{(A, B) \mid A, B \in N \cup \Sigma,\ A \overset{*}{\underset{G}{\Longrightarrow}} B\}$$

bekannt sein. Nach 6.27 und 6.31 kann man sie unabhängig vom Eingabewort w, also *vor* der Analyse berechnen und speichern. Während der Analyse sieht man in den abgespeicherten Relationen nach, ob $A \overset{*}{\underset{G}{\Longrightarrow}} \varepsilon$ bzw $A \overset{*}{\underset{G}{\Longrightarrow}} B$. In ähnlicher Weise kann man auch die Relation

$$\{(A, B) \mid A, B \in N \cup \Sigma,\ \exists \gamma \in (N \cup \Sigma)^*\, A \overset{*}{\underset{G}{\Longrightarrow}} B\gamma\}$$

und die Menge $follow(A) :=$

$$\{a \in \Sigma \mid \exists \alpha, \beta \in (N \cup \Sigma)^*\ \exists u, v \in \Sigma^*\ (S \overset{*}{\underset{G}{\Longrightarrow}} \alpha A \beta\ \wedge\ \alpha A \overset{*}{\underset{G}{\Longrightarrow}} u\ \wedge\ \beta \overset{*}{\underset{G}{\Longrightarrow}} av)\}$$

vorberechnen. Diese werden in 9.12 zur Optimierung der Initialisierung, des Completers und des Predictors verwendet. Um den Predictor weiter zu beschleunigen — auf Kosten des Speicherplatzes! — , könnte man noch die Mengen

$$P(C, a) := \{A \to \alpha \cdot B\beta \mid \exists \gamma\, (C \overset{*}{\underset{G}{\Longrightarrow}} A\gamma),\ (A \to \alpha B \beta) \in P,\ \alpha \overset{*}{\underset{G}{\Longrightarrow}} \varepsilon,\ \exists \delta (B \overset{*}{\underset{G}{\Longrightarrow}} a\delta)\}$$

im voraus berechnen und bei der Analyse die im Predictor zu erzeugenden partiellen Bäume aus den gespeicherten Mengen $P(C, a)$ aussondern.

Algorithmus 9.12 Earley-Analyse mit Aufbau von kanonischen Bäumen

Eingabe: Eine Grammatik $G = (N, \Sigma, P, S)$, ein Wort $w = a_1 \ldots a_n \in \Sigma^+$, und $E \subseteq N$.

Ausgabe: Baummengen $ptrees(i, j, A \to \alpha \cdot \beta)$ und $ctrees(i, j, A)$ wie in 9.13

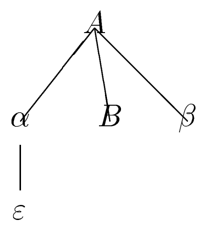
Verfahrensweise:

Initialisierung: $expect(0) := E, \quad expect(j+1) := \emptyset, \quad ctrees(i, j, A) := \emptyset$

Abschlußoperationen: Für $j = 0, 1, 2, \ldots$

$predict(j)$:

$$\left.\begin{array}{l} C \in expect(j), \\ C \overset{*}{\underset{G}{\Longrightarrow}} A\gamma \\ (A \to \alpha B \beta) \in P, \\ \alpha \overset{*}{\underset{G}{\Longrightarrow}} \varepsilon, \\ B \overset{*}{\underset{G}{\Longrightarrow}} a_{j+1}\delta, \\ \beta \neq \varepsilon \end{array}\right\} \Longrightarrow$$

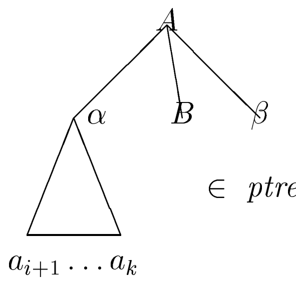


$\in \; ptrees(j, j, A \to \alpha \cdot B\beta)$,
$B \in expect(j)$

$scan(j+1)$: $\quad \cdot\, a_{j+1} \;\in\; ctrees(j, j+1, a_{j+1})$

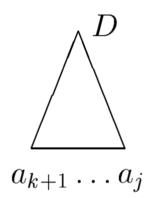
$complete(i, k, j)$: a)

$0 \leq i < k < j \leq n$,



$\in \; ptrees(i, k, A \to \alpha \cdot B\beta)$,

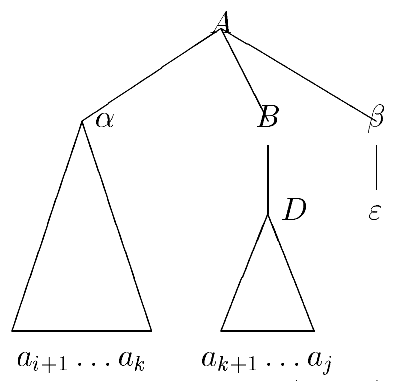
$B \overset{*}{\underset{G}{\Longrightarrow}} D$,



$\in \; ctrees(k, j, D)$

$\beta \overset{*}{\underset{G}{\Longrightarrow}} \varepsilon$,
$a_{j+1} \in follow(A)$,

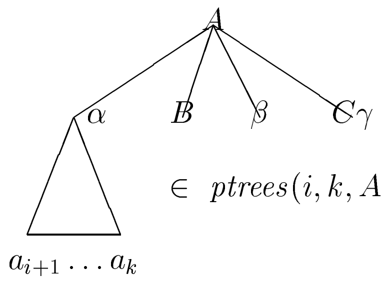
$\Longrightarrow$



$\in \; ctrees(i, j, A)$.

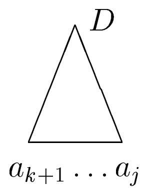
complete(i,k,j): b)

$0 \leq i \leq k < j < n$,



$\in$ *ptrees*$(i,k,A \to \alpha \cdot B\beta C\gamma)$,

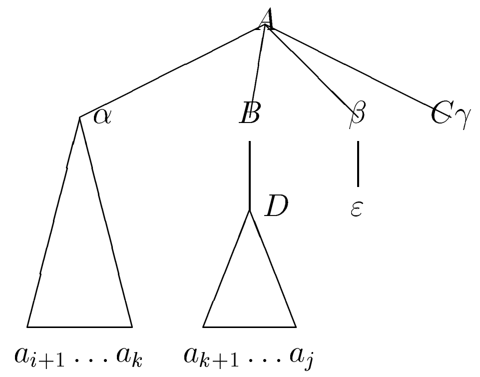
$B \xRightarrow[G]{*} D$,



$\in$ *ctrees*(k,j,D)

$\beta \xRightarrow[G]{*} \varepsilon$,
$C \xRightarrow[G]{*} a_{j+1}\delta$,

$\Longrightarrow$



$\in$ *ptrees*$(i,j,A \to \alpha B\beta \cdot C\gamma)$,
$C \in$ *expect*(j).

Die hiermit erzeugten Baummengen kann man wie folgt charakterisieren:

Satz 9.13 Seien *ptrees*$(i,j,A \to \alpha \cdot B\beta)$ und *ctrees*(i,j,D) mit $A \to \alpha B\beta \in P$ und $D \in N \cup \Sigma$ die von den obigen Abschlußoperationen zu $w = a_1 \ldots a_n$ relativ zu *expect*$(0) := \{S\}$ erzeugten G-Ableitungsbäume.

(i) $t \in$ *ptrees*$(i,j,A \to \alpha \cdot B\beta)$ genau dann, wenn $0 \leq i \leq j$ mit $\beta \neq \varepsilon$ für $i = j$, und wenn es kanonische Linksableitungen $S \xRightarrow[G]{*} a_1 \ldots a_i A\theta$, $\alpha \xRightarrow[G]{*} a_{i+1} \ldots a_j$, und $B \xRightarrow[G]{*} a_{j+1}\eta$ gibt und t der Ableitungsbaum zu

$$A \xRightarrow[G]{} \alpha B\beta \xRightarrow[G]{*} a_{i+1} \ldots a_j B\beta$$

ist.

(ii) $t \in$ *ctrees*(i,j,D) genau dann, wenn entweder $i+1 = j$, $D = a_{i+1}$ und t der einpunktige mit a_{i+1} markierte Baum ist, oder wenn es k mit $i < k < j$ und kanonische Linksableitungen $S \xRightarrow[G]{*} a_1 \ldots a_i D\delta$ und

$$\begin{aligned} D \xRightarrow[G]{} \alpha B\beta &\xRightarrow[G]{*} a_{i+1} \ldots a_k B\beta \\ &\xRightarrow[G]{*} a_{i+1} \ldots a_k a_{k+1} \ldots a_j \beta \\ &\xRightarrow[G]{*} a_{i+1} \ldots a_k a_{k+1} \ldots a_j \varepsilon \end{aligned}$$

gibt und t der G-Ableitungsbaum der letzteren Ableitung ist.

Aus dieser Charakterisierung folgt, daß

$$
parse(a_1 \dots a_n, G, E) := \left\{ \begin{array}{c} A \\ | \\ D \\ \triangle \end{array} \;\middle|\; A \in E,\ A \overset{*}{\underset{G}{\Longrightarrow}} D,\ \begin{array}{c} D \\ \triangle \end{array} \in \; ctrees(0, n, D) \right\}
$$

alle Analysebäume (mit nur kanonischen Kettenableitungen und Löschungen) der Eingabe $w = a_1 \dots a_n$ enthält, deren Wurzel mit einem $A \in E$ markiert ist.

Auf die Beweise wird hier verzichtet. Im Fall a) des Completers verzweigt der erzeugte Baum wegen $i < k < j$ an der Spitze in mindestens zwei Teilbäume, deren Fronten nichtleere (!) Teilworte der Eingabe enthalten. Im Fall b) kann wegen $i \leq k < j$ zwar $a_{i+1} \dots a_k = \varepsilon$ sein, aber nicht $a_{k+1} \dots a_j$. Fortsetzungen von $t \in ptrees(i, j, A \to \alpha \cdot B\beta)$ können daher das Nonterminal B nicht vollständig löschen.

Bemerkung 9.14 (i) Das Verfahren von Earley ist eine Mischung aus Top-Down- und Bottom-Up-Vorgehen. Vom Top-Down-Ansatz übernimmt man die Erwartungen, die — in der Voraussetzung des Predictors — gewisse Strukturen aussondern. Der eigentliche Aufbau komplexer Strukturen erfolgt durch den Completer in Bottom-Up-Weise.

(ii) Man sieht an der ursprünglichen Form aus 9.10, daß der Scanner als Spezialfall des Completers betrachtet werden kann: dazu läßt man Pseudoregeln $a \to a$ für $a \in \Sigma$ zu und gibt

$$[a_j \to a_j \cdot, j-1] \in Items_j$$

in der Initialisierungsphase (oder im Scanner) vor. Dem entspricht in 9.12, daß wir $\cdot\, a_j$ als vollständigen Baum und $\begin{array}{c} a \\ | \\ a \end{array}$ als kanonische Ableitung erlauben.

(iii) Man beachte, daß der Predictor nur Strukturen der Tiefe 1 aufbaut. Deswegen läuft man auch bei Linksrekursionen nicht in eine endlose Schleife: ein Baum wie

$$\begin{array}{ccc} & A & \\ / & & \backslash \\ A & & \beta \end{array}$$

wird nur einmal in $ptrees(j, j, A \to \cdot A\beta)$ erzeugt, aber nicht etwa ergänzt zu

$$\begin{array}{ccccc} & & A & & \\ & / & & \backslash & \\ & A & & \beta & \\ / & & \backslash & & \\ A & & \beta & & \end{array} .$$

Wie in 9.10 kann man die Bildung des Abschlusses unter den in in 9.12 angegebenen Regeln deterministisch durch eine geschachtelte Schleife erreichen. Dabei ist es günstig, $scan(j+1)$ vor $complete(j)$ auszuführen, so daß man direkt auf die Vorberechnungen zugreifen kann. Im Unterschied zu 9.10 braucht man hier keine Information über die Vorausschau zu speichern, um den Filtereffekt einer Vorausschau um ein Eingabesymbol zu erreichen.

Aufgaben

Aufgabe 9.1 a) Man gebe den Top-Down- oder den Bottom-Up-Analyseautomaten (mit Ausgabe von Regelnummern) für die Grammatik aus Aufgabe 8.1 an.

b) Man gebe für die Nominalphrase aus Aufgabe 8.1 einen akzeptierenden Lauf des Automaten aus a) an und notiere die ausgegebenen Regelnummern am Analysebaum.

Aufgabe 9.2 a) Man gebe zur Grammatik $G = (\{a, b, c\}, \{A, S\}, P, S)$ mit der Regelmenge

$$P = \begin{cases} 1\text{: S} \to \text{aAS} & 2\text{: S} \to \text{bA} \\ 3\text{: A} \to \text{bSb} & 4\text{: A} \to \text{cA} \\ 5\text{: A} \to \text{a} & \end{cases}$$

den nicht-deterministischen top-down-Analyse-Automaten M^G und den deterministischen top-down-Analyse-Automaten $M^{G,look}$ “mit lookahead” an.

Hinweis: Für $M^{G,look}$ ersetze man (wie in Abschnitt 9.2) den *predict*-Teil durch *predict*$\,'$, was aus

$$check \circ predict \circ lookahead \qquad (lookahead = \{\, (\epsilon, a \cdot a^{-1}) \mid a \in \Sigma \,\})$$

entsteht, indem man alle nicht ausführbaren Operationen der Form $(\ldots a^{-1} b \ldots, \ldots)$ mit Terminalen $a \neq b$ wegläßt.

b) Man gebe zu der leicht veränderten Grammatik G' mit den Regeln

$$P' = \left\{ \begin{array}{ll} 1\text{: S} \rightarrow \text{aAS} & 2\text{: S} \rightarrow \text{Ab} \\ 3\text{: A} \rightarrow \text{bSb} & 4\text{: A} \rightarrow \text{cA} \\ 5\text{: A} \rightarrow \text{d} & \end{array} \right.$$

ebenfalls einen deterministischen top-down-Analyse-Automaten mit lookahead an.

Hinweis: Hierbei muß man aber *predict'* über

$$\mathit{check} \circ \mathit{predict} \circ \mathit{predict} \circ \mathit{lookahead}$$

berechnen, da die rechten Seiten der S-Regeln noch nicht mit paarweise verschiedenen Terminalen beginnen.

c) Man verändere die verwendeten Operationen so, daß mit einer dritten Komponente die Nummern der jeweils verwendeten Regeln ausgegeben werden. Die Operationen haben dann die Form

(Kellerveränderung, Eingabeveränderung, verwendete Regelnummer(n)).

Zeige an den Wörtern *baddbbb* und *acdd*, wie die Abarbeitung vor sich geht.

Aufgabe 9.3 Man gebe zu den Eingaben den Eingabe $w_1 = ababbcb$ und $w_2 = bbabbbc$

(i) zur Grammatik G (in Chomsky-Normalform) mit den Regeln

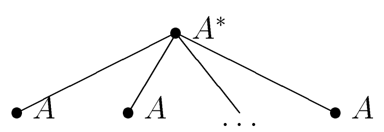
$$P = \left\{ \begin{array}{ll} 1\text{: S} \rightarrow \text{AS} & 2\text{: S} \rightarrow \text{BA} \\ 3\text{: A} \rightarrow \text{BE} & 4\text{: A} \rightarrow \text{CA} \\ 5\text{: A} \rightarrow \text{a} & 6\text{: B} \rightarrow \text{b} \\ 7\text{: C} \rightarrow \text{c} & 8\text{: D} \rightarrow \text{AS} \\ 9\text{ A} \rightarrow \text{AA} & 10\text{: E} \rightarrow \text{SB} \end{array} \right.$$

die Analysetabelle nach dem Algorithmus von Cocke-Younger-Kasami und

(ii) zur Grammatik G aus Aufgabe 9.2 die Item-Tabellen nach dem Earley-Algorithmus

an und stelle damit fest, ob $w_i \in L(G)$ ist.

Aufgabe 9.4 Man erweitere die Abschlußregeln in Algorithmus 9.12 so, daß man für Iterationen nicht links- oder rechtsrekursiv untergeordnete, sondern sondern wie in



nebengeordnete Strukturen aufbaut. Dies kann z.B. bei der Analyse von mehrgliedrigen Konjunktionen in natürlichen Sprachen sinnvoll sein.

10 Varianten kontextfreier Grammatiken

In diesem Kapitel werden kurz zwei Varianten kontextfreier Grammatiken skizziert, die in neuerer Zeit in der Linguistik verwendet werden: die Baumadjunktionsgrammatiken und eine stärker logisch als algebraisch organisierte Art von Grammatiken, die Kategorialgrammatiken.

10.1 Baumadjunktionsgrammatiken

Die Grundidee des Satzes 8.12 von R. Parikh bestand darin, Ableitungsbäume aus minimalen Basisbäumen mit einer Front aus Terminalsymbolen und minimalen "Rekursionsbäumen", deren Front außer Terminalsymbolen noch ein Vorkommen der Wurzelmarke enthält, zusammenzusetzen.

A. Joshi hat diese Idee des Aufpumpens von Bäumen einer kontextfreien Grammatik etwas verallgemeinert: die Basisbäume und die Rekursionsbäume müssen nicht mehr Ableitungsbäume einer kontextfreien Grammatik sein, sondern können aus sonstwie vorgegebenen endlichen Baummengen stammen.

Definition 10.1 Eine Baumadjunktionsgrammatik $G = (N, \Sigma, \mathcal{B}, \mathcal{C}, S)$ (oder TAG, tree-adjoining-grammar) besteht aus endlichen Mengen N, Σ mit $N \cap \Sigma = \emptyset$, und endlichen Mengen $\mathcal{B}$ und $\mathcal{C}$ von markierten geordneten endlichen Bäumen, so daß gilt:

(i) Für jeden Basisbaum $(t, \prec, m) \in \mathcal{B}$ ist $m(0) = S$, $m(x) \in \Sigma$ für Blätter x von t, und $m(x) \in N$ für innere Knoten x von t.

(ii) Für jeden Rekursionsbaum $(t, \prec, m) \in \mathcal{C}$ ist $m(x) \in N$ für innere Knoten x von t, $m(x) \in \Sigma$ für alle bis auf eines der Blätter x von t, und $m(x) = m(0) \in N$ für das verbleibende Blatt x.

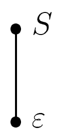
Die von G erzeugte Sprache über Σ ist

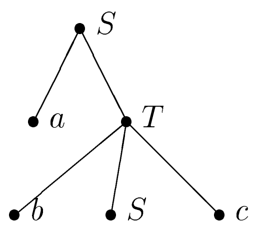
$$L(G) = \left\{ w \in \Sigma^* \;\middle|\; \begin{array}{c} \text{Es gibt einen aus einem Basisbaum } \tau_o \in \mathcal{B} \text{ durch "Einhängen"} \\ \text{von } \mathcal{C}\text{-Bäumen entstandenen Baum } \tau \text{ mit Front}(\tau) = w \end{array} \right\}$$

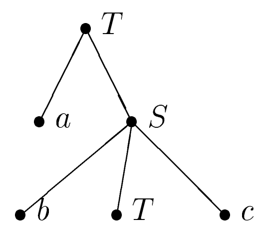
Eine Sprache $L \subseteq \Sigma^*$ heißt Baumadjunktionssprache, (oder TAL, tree-adjoining-language), falls es eine Baumadjunktionsgrammatik G mit $L = L(G)$ gibt.

Für die Operation des Einhängens wird auf den Beweis von 6.11 verwiesen.

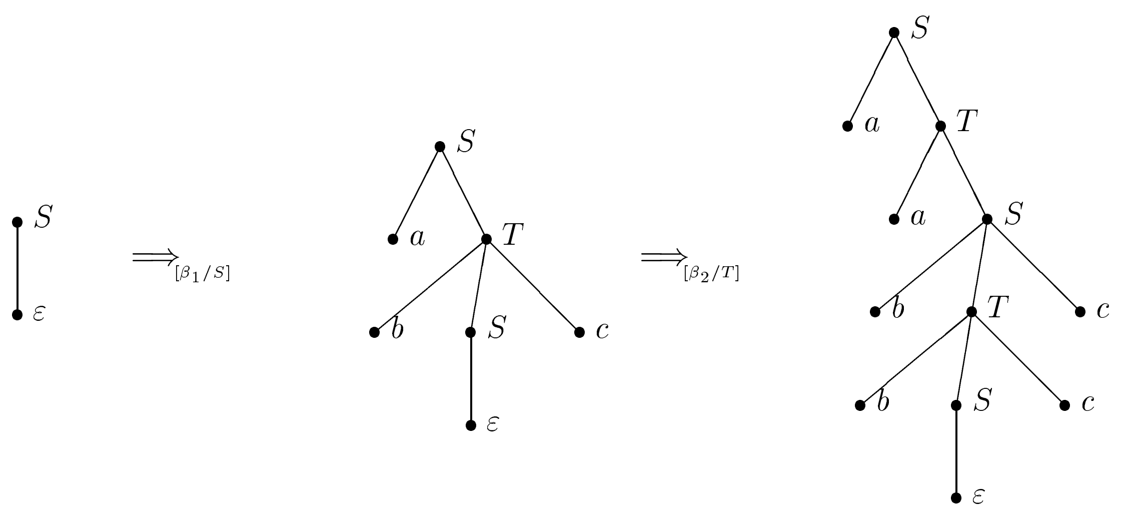
Beispiel 10.2 Sei $G = (N, \Sigma, \mathcal{B}, \mathcal{C}, S)$ mit $N = \{S, T\}$, $\Sigma = \{a, b, c\}$, $\mathcal{B} = \{\alpha\}$ und $\mathcal{C} = \{\beta_1, \beta_2\}$, wobei

$\alpha :=$ 

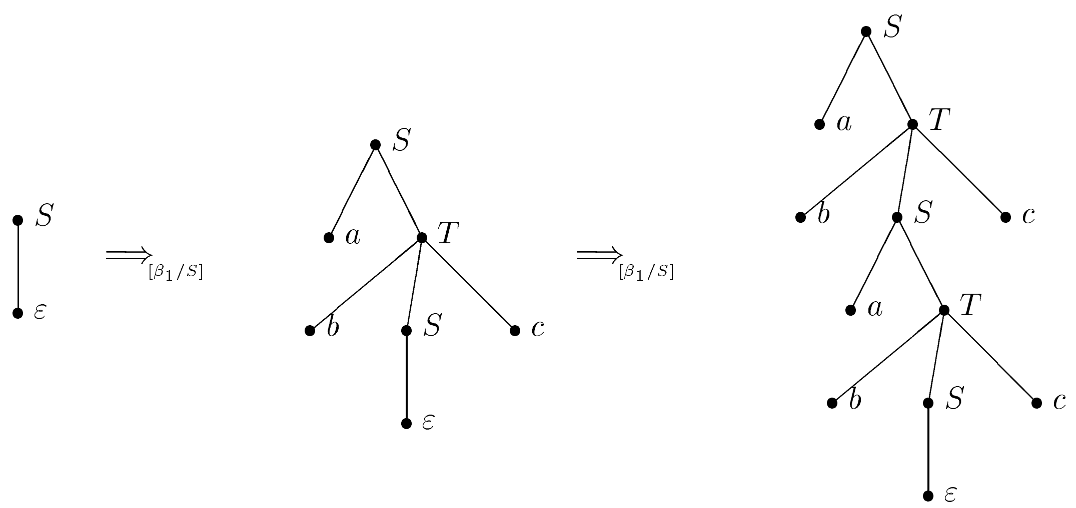
$\beta_1 :=$ 

$\beta_2 :=$ 

Durch Einhängen kann man aus α z.B. erzeugen:



aber auch



Daher sind $\varepsilon, abc, aabbcc, ababcc \in L(G)$.

Satz 10.3 Das Parikh-Bild $||L||$ einer Baumadjunktionssprache L ist semilinear.

Das zeigt man mit demselben Beweis wie für Satz 6.11. Der einzige Unterschied besteht darin, daß hier weniger strenge Einschränkungen an die Basisbäume und die Hilfsbäume gemacht werden. Dieser Unterschied bewirkt aber:

Satz 10.4 (Joshi 1986) Die Klasse der kontextfreien Sprachen ist eine ECHTE Teilklasse der Baumadjunktionssprachen.

Beweis: (Skizze) Die Inklusion $\subseteq$ ist klar nach dem Beweis von 5.11. Für die Verschiedenheit benutzt man die Grammatik G aus Beispiel 9.2, die die Sprache $L(G) =$

$$\{\, wc^n \mid n \geq 0,\ w \in \{a,b\}^*,\ ||w|| = (n,n),\ \text{für jedes Präfix } v \text{ von } w \text{ ist } |v|_a \geq |v|_b \,\}$$

definiert. $L(G)$ ist nach 8.10 nicht kontextfrei, da

$$L(G) \cap a^*b^*c^* = \{\, a^nb^nc^n \mid n \geq 0 \,\},$$

der Schnitt von $L(G)$ mit einer regulären Menge, nicht kontextfrei ist. □

10.2 Kategorialgrammatiken

In den von Ajdukiewicz in den 30er Jahren eingeführten Kategorialgrammatiken versucht man, die Wahl der syntaktischen Kategorien durch semantische Kategorien zu rechtfertigen: Aussagesätze sind dann Phrasen, deren Bedeutung ein Wahrheitswert ist, Nominalphrasen sind Ausdrücke, deren Bedeutung ein Objekt ist, Verben sind Ausdrücke, deren Bedeutung eine Funktion ist, die Objekten Wahrheitswerte zuordnet.

Setzt man als syntaktische Grundkategorien Sätze S und Nomina N an, so kann man die Kategorie der Verben V oder Verbalphrasen als von diesen Grundkategorien *abhängige* Kategorie ansehen: zusammen mit einer Nominalphrase ergibt eine Verbalphrase einen Satz. Diese Abhängigkeit soll in der Bezeichnung der Kategorie zum Ausdruck kommen, und berücksichtigt man noch, ob das Nomen vor oder nach dem Verb steht, so kann man zwei syntaktische Kategorien $N\backslash S$ und S/N statt V einführen. Statt einer kontextfreien Regel wie $S \longrightarrow NV$ hat man dann $N \cdot N\backslash S \subseteq S$, wenn man die Kategoriensymbole durch Wortmengen interpretiert.

Jedem Wort w der Sprache werden nun syntaktische Kategorien zugeordnet, etwa

$$\text{Peter} : N, \quad \text{lacht} : N\backslash S, \quad \text{selten} : S\backslash S.$$

Für zusammengesetzte Ausdrücke kann man dann eine syntaktische Kategorie berechnen, z.B.

$$\text{Peter lacht} : N \cdot (N\backslash S) \subseteq S, \quad (\text{Peter lacht}) \text{ selten} : S \cdot (S\backslash S) \subseteq S.$$

Einem Wort können mehrere Kategorien zugeordnet sein, und bei manchen Wörtern, deren Bedeutung Funktionen sind, müssen die Ausdrücke, die den Argumenten der Funktion entsprechen, teilweise *vor* und teilweise *nach* dem Wort stehen, z.B.

$$\text{und} : S\backslash(S/S), \qquad \text{und} : N\backslash(N/N).$$

Dieses Vorgehen motiviert die folgenden Definitionen.

Definition 10.5 Sei $B_0, B_1, \ldots$ eine abzählbare Menge. Die Menge Cat der syntaktischen Kategorien ist die kleinste Menge mit (i) jede Basiskategorie B_i, $i \in I\!N$, ist eine Kategorie, und (ii) sind A und B Kategorien, so auch $(B\backslash A)$ und (A/B).

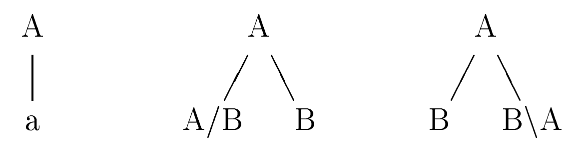
Eine Kategorialgrammatik $K = (\Sigma, T, S)$ besteht aus einem endlichen Alphabet Σ, einer Funktion T, die jedem $a \in \Sigma$ eine endliche Teilmenge $T(a) \subseteq$ Cat zuordnet, und einer ausgezeichneten Kategorie $S \in$ Cat.

Ein Wort $w \in \Sigma^+$ ist von der Kategorie A, in Zeichen: $w : A$, wenn (i) $w \in \Sigma$ und $A \in T(w)$, oder (ii) es gibt eine Kategorie B und Wörter $u : A/B$, $v : B$, so daß $w = uv$, oder (iii) es gibt eine Kategorie B und Wörter $u : B$, $v : (B\backslash A)$, so daß $w = uv$. Die von K definierte Sprache ist $L(K) := \{w \in \Sigma^* \mid w : S\}$.

Wir sprechen von Ableitungsbäumen bezüglich K, indem wir die Bedingungen (i) bis (iii) als Produktionsregeln

$$\begin{array}{ll} A \to a, & \text{mit } A \in T(a), a \in \Sigma \\ A \to B \cdot (B\backslash A), & A, B \text{ Kategorien von } K, \\ A \to (A/B) \cdot B, & A, B \text{ Kategorien von } K \end{array}$$

auffassen und endliche Bäume aus den entsprechenden Stücken



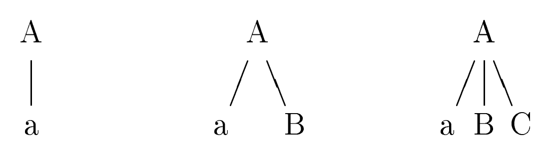
zusammensetzen. Beachte aber, daß wir im Unterschied zu kontextfreien Grammatiken hier unendlich viele (zusammengesetzte) "Nonterminale" haben. Wie man leicht sieht, gibt es zu jeder Kategorialgrammatik eine äquivalente kontextfreie Grammatik. Schwieriger ist die Umkehrung:

Satz 10.6 (Gaifman 1957) Zu jeder kontextfreien Grammatik $G = (N, \Sigma, P, S)$ mit $\varepsilon \notin L(G)$ gibt es eine äquivalente Kategorialgrammatik $K = (N, \Sigma, T, S)$.

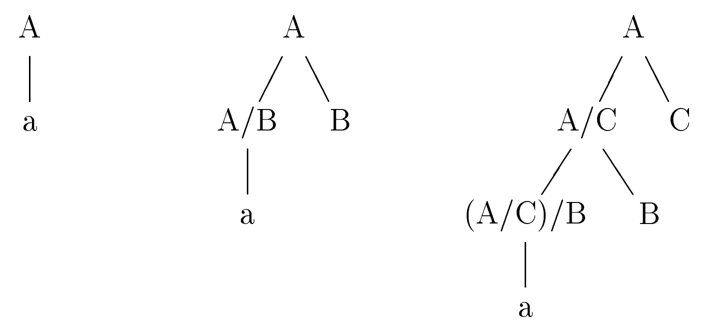
Beweis: Nach Satz 6.17 können wir annehmen, daß G in Greibach–Normalform gegeben ist, so daß alle Regeln von der Form $A \to a\beta$ mit $a \in \Sigma$ und $\beta \in (N \cup \Sigma)^*$ sind. Zur Vereinfachung der Schreibweise nehmen wir im Folgenden zusätzlich an, daß β aus höchstens zwei Nonterminalen besteht. Für $a \in \Sigma$ sei dann

$$\begin{aligned} T(a) := \; & \{\, A \mid (A \to a) \in P \,\} \cup \\ & \{\, A/B \mid (A \to aB) \in P \,\} \cup \\ & \{\, (A/C)/B \mid (A \to aBC) \in P \,\}. \end{aligned}$$

Alle Ableitungsbäume von G sind aus Bäumen der Formen



zusammengesetzt, denen folgende Bäume bezüglich K entsprechen:



Diese Zuordnung erzeugt eine Abbildung

$$\kappa : G\text{-Ableitungsbäume} \to K\text{-Ableitungsbäume}.$$

Da wir die Menge N der Nonterminalsymbole von G als Basiskategorien von K gewählt haben, gehören die bei dieser Korrespondenz nötigen Kategorien zu den Kategorien von K. Da die Marken an den Wurzeln und Blättern der korrespondierenden Bäume gleich sind, gilt offenbar:

Beh. 1: Ein G-Ableitungsbaum τ und der zugeordnete K-Ableitungsbaum $\kappa(\tau)$ haben die gleiche Wurzel- und Blättermarkierung.

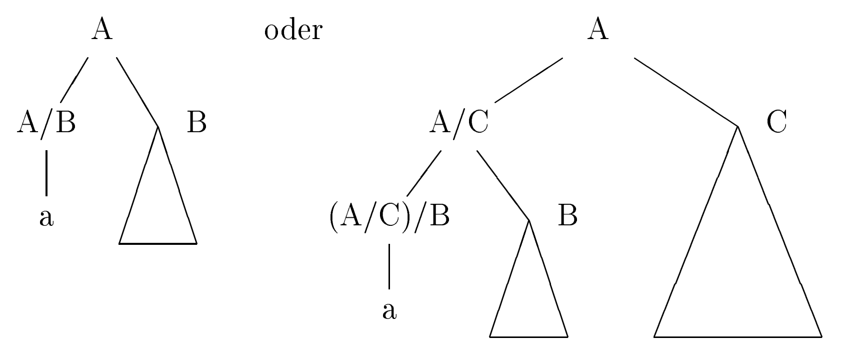
Daraus folgt sofort $L(G) \subseteq L(K)$. Für die Umkehrung genügt es zu zeigen:

Beh. 2: Zu jedem K-Ableitungsbaum ϱ mit Blättermarken in $\Sigma \cup N$ und Wurzelmarke in N gibt es einen G-Ableitungsbaum τ mit $\varrho = \kappa(\tau)$.

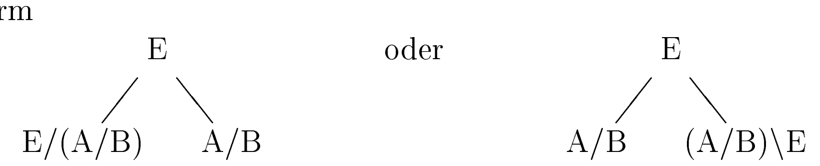
Dies sieht man durch Induktion über die Anzahl der Knoten von ϱ. Falls ϱ nur zwei Knoten hat, ist die Behauptung klar, da der Baum von einer Regel $A \to a$ stammt. Im anderen Fall gibt es einen verzweigenden Knoten, da jeder nichtverzweigende Knoten in einem Terminalsymbol endet.

Jeder verzweigende Knoten hat einen direkten Nachfolger, der mit einer zusammengesetzten Kategorie markiert ist. Dieser Nachfolger kann nach der Voraussetzung über die Blättermarkierung kein Blatt sein. Da ϱ endlich ist, gibt es mindestens einen Knoten, der mit einer zusammengesetzten Kategorie markiert, aber kein Verzweigungsknoten ist. Wie man an den K-Ableitungsbäumen der Tiefe 1 sieht, dominiert er ein Blatt, das mit einem Terminalsymbol $a \in \Sigma$ markiert ist.

Nach den Kategorisierungsannahmen über a muß ϱ daher einen Teilbaum der Form



haben, wobei A, B und C Basiskategorien sind. Das liegt daran, daß alle nicht mit Terminalsymbolen markierten Blätter von ϱ mit Basiskategorien D markiert sind und alle mit einem Terminalsymbol b markierten Blätter von einem Knoten mit einer Markierung aus $T(b)$ dominiert werden, also einer der Kategorien C_1, C_1/C_2, oder $(C_1/C_2)/C_3$ mit Basiskategorien C_i. Daher kann A/B — und entsprechend $(A/B)/C$ — nicht "als Argument", d.h. in der Form



verwendet werden, sondern nur "als Funktion" wie oben behauptet.

Nach der Induktionsannahme sind die von B bzw. C dominierten Teilbäume von ϱ schon die κ-Bilder von G-Ableitungsbäumen. Das Anfangsstück von ϱ, das man durch Entfernen dieser Teilbäume (bis auf ihre Wurzeln B und C) erhält, ist ebenfalls das κ-Bild eines G-Ableitungsbaums. Setzt man diese G-Ableitungsbäume mit dem G-Baum zur Regeln $A \to aB$ oder $A \to aBC$ zusammen, so erhält man einen G-Ableitungsbaum τ mit $\varrho = \kappa(\tau)$. □

Die hier beschriebenen Kategorialgrammatiken wurden durch Lambek[13] erweitert und bilden gegenwärtig eine wesentliche Forschungsrichtung in der theoretischen Linguistik.

Lambek-Grammatiken

In einer Kategorialgrammatik wird die Feststellung, ob ein Ausdruck w ein Satz ist, darauf zurückgeführt, daß man $w : A$ und $A \subseteq S$ für eine geeignete Kategorie A zeigt. Dabei wird für Kategorien C_1 und C_2 die Aussage $C_1 \subseteq C_2$ durch mehrfaches Anwenden der "Kürzungsregeln"

$$B \cdot (B\backslash A) \subseteq A \qquad \text{und} \qquad (A/B) \cdot B \subseteq A$$

gezeigt. J. Lambek[13] hat einen logischen Kalkül vorgeschlagen, der weitere Möglichkeiten erlaubt, solche Aussagen $C_1 \subseteq C_2$ zu beweisen.

Definition 10.7 Der SYNTAKTISCHE KALKÜL VON LAMBEK hat folgende Ausdrücke, Axiome und Regeln.

(i) Die Ausdrücke sind die aus einer abzählbaren Menge $B_0, B_1, \ldots$ von atomaren Kategorien durch (A/B), $(A\backslash B)$ und $(A \cdot B)$ erzeugte Menge von KATEGORIEN. Eine SEQUENZ $C_1, \ldots, C_n \vartriangleright C_0$ ist eine Folge von Kategorien C_i mit $n \geq 1$.

(ii) AXIOME sind alle Sequenzen $A \vartriangleright A$, für atomare Kategorien A.

(iii) SCHLUSSREGELN zur Herleitung von Sequenzen sind folgende, wobei Γ, Δ und Θ für endliche Folgen von Kategorien stehen, und A, B für beliebige Kategorien :

$$\frac{\Theta \vartriangleright A \qquad \Gamma, B, \Delta \vartriangleright C}{\Gamma, \Theta, A\backslash B, \Delta \vartriangleright C}\ (\backslash\ L) \qquad\qquad \frac{A, \Gamma \vartriangleright B}{\Gamma \vartriangleright A\backslash B}\ (\backslash\ R)$$

$$\frac{\Theta \vartriangleright A \qquad \Gamma, B, \Delta \vartriangleright C}{\Gamma, B/A, \Theta, \Delta \vartriangleright C}\ (/\ L) \qquad\qquad \frac{\Gamma, A \vartriangleright B}{\Gamma \vartriangleright B/A}\ (/\ R)$$

$$\frac{\Gamma, A, B, \Delta \vartriangleright C}{\Gamma, A \cdot B, \Delta \vartriangleright C}\ (\cdot\ L) \qquad\qquad \frac{\Gamma \vartriangleright A \qquad \Delta \vartriangleright B}{\Gamma, \Delta \vartriangleright A \cdot B}\ (\cdot\ R)$$

$$\frac{\Theta \vartriangleright B \qquad \Gamma, B, \Delta \vartriangleright A}{\Gamma, \Theta, \Delta \vartriangleright A}\ (\mathit{Schnitt})$$

Wir schreiben $\Gamma \vdash A$, wenn $\Gamma \rhd A$ mit diesen Regeln in endlich vielen Schritten aus den Axiomen herleitbar ist. Eine Lambek'sche Kategorialgrammatik $G = (\Sigma, T, S)$ ordnet der Kategorie A die Wortmenge

$$A^G := \{\, a_1 \cdots a_n \mid n \in \mathbb{N},\ A_1 \in T(a_1), \ldots, A_n \in T(a_n),\ A_1, \ldots, A_n \vdash A \,\}$$

zu. $L(G) := S^G$ ist DIE VON G DEFINIERTE SPRACHE.

Mit den Regeln ($\backslash\, R$) und ($/\, R$) kann man zum Nachweis der Inklusion $C_1 \cdots C_n \subseteq C_0$ auch Hilfsannahmen verwenden, z.B. bei ($/\, R$): wenn beliebige Ausdrücke v der Kategorie $C_1 \cdots C_n$, *unter der Annahme*, es folgt ein Ausdruck w der Kategorie A, stets Ausdrücke vw der Kategorie B ergeben, dann sind die Ausdrücke der Kategorie $C_1 \cdots C_n$ von der Kategorie B/A.

Lemma 10.8 (Lambek 1957) (i) Jede herleitbare Sequenz kann auch ohne Verwendung der Schnittregel hergeleitet werden. (ii) Es gibt ein Verfahren, mit dem man für jede Sequenz $\Gamma \rhd A$ feststellen kann, ob sie herleitbar ist.

Proposition 10.9 Die durch $G = (\Sigma, T, S)$ definierte Sprachfamilie $\{\, A^G \mid A \in \textsc{Cat} \,\}$ ist die Familie der kleinsten Sprachen $[\![A]\!] \subseteq \Sigma^+$ mit

$$A^T := \{\, a \in \Sigma \mid A \in T(a) \,\} \subseteq [\![A]\!] \quad \text{und} \quad [\![A_1]\!] \cdots [\![A_n]\!] \subseteq [\![A]\!], \text{ falls } A_1, \ldots, A_n \vdash A.$$

Beweis: (Skizze) Man kann in 2^{Σ^+} ja das unendliche System aller Gleichungen

$$A = A^T \cup \bigcup \{\, A_1 \cdots A_n \mid n \in \mathbb{N},\ A_1, \ldots, A_n \in \textsc{Cat},\ A_1, \ldots, A_n \vdash A \,\}$$

wie im Fixpunktsatz durch Approximation von unten lösen. □

Im Unterschied zu kontextfreien Grammatiken werden hier die Bedeutungen *unendlich vieler* Kategorien simultan induktiv definiert. Nach dem folgenden Satz, der eine Vermutung von Chomsky aus den 60er Jahren bestätigt, kann man sich auf endlich viele Kategorien beschränken:

Satz 10.10 (Pentus 1992) Zu jeder Lambek-Grammatik G gibt es eine kontextfreie Grammatik G' mit $L(G) = L(G')$.

Beweis: (Skizze) Sei $|C|$ die Länge der Kategorie C, definiert durch

$$|C| := \begin{cases} 1, & \text{falls } C \text{ atomar ist,} \\ |A| + |B|, & \text{falls } C \in \{A/B, B\backslash A, A \cdot B\}. \end{cases}$$

Sei $m := max\{\,|C| \mid C = S$ oder $C \in T(a)$ für ein $a \in \Sigma\}$. Für $G' = (N, \Sigma, P, S)$ wähle

$$\begin{array}{rcl} N & := & \{\, C \in \textsc{Cat} \mid |C| \leq m \,\}, \\ P & := & \{\, A \to a \mid A \in T(a) \,\} \\ & \cup & \{\, A_0 \to A_1 \ldots A_k \mid 1 \leq k \leq 2,\ A_0, \ldots, A_k \in N,\ A_1, \ldots, A_k \vdash A_0 \,\}. \end{array}$$

Beh. 1: $L(G) \subseteq L(G')$:

Sei $a_1 \cdots a_n \in L(G)$. Dann gibt es $A_i \in T(a_i)$ mit $A_1, \ldots, A_n \vdash S$. Nach Wahl von G' ist $A_i \to a_i \in P$, für jedes $i \leq n$. Es genügt also zu zeigen, daß $S \Rightarrow^*_{G'} A_1 \cdots A_n$. Sei

$$Ax_{2,m} := \{A_1, \ldots, A_k \ \triangleright\ A_0 \mid 1 \leq k \leq 2,\ A_0, \ldots, A_n \in N,\ A_1, \ldots, A_k \vdash A_0\}.$$

Hilfssatz 10.11 Für jede Sequenz $A_1, \ldots, A_n \triangleright A_0$ sind äquivalent: (i) $A_1, \ldots, A_n \vdash A_0$ und $|A_i| \leq m$ für alle $i \leq n$, (ii) $A_1, \ldots, A_n \triangleright A_0$ ist aus $Ax_{2,m}$ allein mit der Schnittregel herleitbar.

Der Beweis dieses Hilfssatzes ist schwierig (siehe Pentus[17]) und wird hier weggelassen. Aus dem Hilfssatz folgt, daß $A_1, \ldots, A_n \triangleright S$ aus $Ax_{2,m}$ allein mit der Schnittregel hergeleitet werden kann. Dabei können wir weiter annehmen, daß in dieser Ableitung die linke Annahme $\Theta \triangleright B$ einer Regelanwendung

$$\frac{\Theta \ \triangleright\ B \qquad \Gamma, B, \Delta \ \triangleright\ A}{\Gamma, \Theta, \Delta \ \triangleright\ A}\ (Schnitt)$$

stets eine Sequenz aus $Ax_{2,m}$ ist: sonst ersetzen wir ein Beweisstück

$$\frac{\dfrac{\Phi \ \triangleright\ B \qquad \Gamma_2, B, \Delta_2 \ \triangleright\ A}{\Gamma_2, \Phi, \Delta_2 \ \triangleright\ A}\ (Schnitt) \qquad \Gamma_1, A, \Delta_1 \ \triangleright\ C}{\Gamma_1, \Gamma_2, \Phi, \Delta_2, \Delta_1 \ \triangleright\ C}\ (Schnitt)$$

durch den Beweis

$$\frac{\Phi \ \triangleright\ B \qquad \dfrac{\Gamma_2, B, \Delta_2 \ \triangleright\ A \qquad \Gamma_1, A, \Delta_1 \ \triangleright\ C}{\Gamma_1, \Gamma_2, B, \Delta_2, \Delta_1 \ \triangleright\ C}\ (Schnitt)}{\Gamma_1, \Gamma_2, \Phi, \Delta_2, \Delta_1 \ \triangleright\ C}\ (Schnitt),$$

wobei der rechte Teilbeweis schon die gewünschte Eigenschaft hat, wie man durch Induktion über die Anzahl der Beweisschritte sieht. Da die linken Annahmen in solchen Herleitungen stets in $Ax_{2,m}$ sind und damit Regeln von P entsprechen, sieht man durch Induktion über die Anzahl der Beweisschritte, daß für so hergeleitete Sequenzen $A_1, \ldots, A_n \triangleright A_0$ in der Tat $A_0 \Rightarrow^*_{G'} A_1 \ldots A_n$ gilt. Damit ist Beh. 1 gezeigt.

Beh. 2: $L(G') \subseteq L(G)$.

Wir setzen die Kategorisierung durch $T(B) := \{B\}$ für $B \in N$ auf die Nonterminale fort. Durch Induktion zeigt man dann für $X_i \in \Sigma \cup N$:

> Falls $S \Rightarrow^*_{G'} X_1 \cdots X_n$, dann gibt es $B_i \in T(X_i)$, so daß $B_1, \ldots, B_n \vartriangleright S$ aus $Ax_{2,m}$ nur mit der Schnittregel herleitbar ist.

Da alle Sequenzen in $Ax_{2,m}$ im Lambek-Kalkül herleitbar sind, folgt für $a_1 \cdots a_n \in L(G')$, daß $a_1 \cdots a_n \in L(G)$ ist. □

Vollständigkeit des Lambek'schen Kalküls

Definition 10.12 Ein Standard- oder Sprachenmodell des Lambek'schen Kalküls ist eine Unterstruktur $\mathcal{M} \subseteq (2^{\Sigma^+}, \cdot, \backslash, /, [\![B_i]\!])_{i \in I\!N}$, wobei Σ ein endliches Alphabet, $[\![B_i]\!] \subseteq \Sigma^+$ eine Sprache über Σ, für jede atomate Kategorie B_i, und $\backslash$ und / die zur Verkettung gehörenden Divisionen sind.

Jeder Kategorie C entspricht in $\mathcal{M}$ eine Sprache $[\![C]\!]^{\mathcal{M}}$, kurz: $[\![C]\!]$, durch

$$\begin{aligned} [\![(A \cdot B)]\!] &:= [\![A]\!] \cdot [\![B]\!], \\ [\![A/B]\!] &:= \{\, v \in \Sigma^+ \mid \{v\} \cdot [\![B]\!] \subseteq [\![A]\!] \,\}, \\ [\![B \backslash A]\!] &:= \{\, v \in \Sigma^+ \mid [\![B]\!] \cdot \{v\} \subseteq [\![A]\!] \,\}. \end{aligned}$$

Eine Sequenz $A_1, \ldots, A_n \vartriangleright A$ heißt gültig in $\mathcal{M}$, falls $[\![A_1]\!] \cdots [\![A_n]\!] \subseteq [\![A]\!]$.

Die beweisbaren Sequenzen sind –zumindest wenn keine mit $\cdot$ gebildeten Kategorien darin vorkommen– genau die in Sprachmodellen allgemeingültigen:

Satz 10.13 (Buszkowski 1982) Sei $\Gamma \vartriangleright A$ eine Sequenz, in deren Kategorien die Verkettung $\cdot$ nicht vorkommt.

(i) (Korrektheit) Ist $\Gamma \vdash A$, so ist $\Gamma \vartriangleright A$ gültig in jedem Standardmodell.

(ii) (Vollständigkeit) Ist $\Gamma \vartriangleright A$ in jedem Standardmodell gültig, so ist $\Gamma \vdash A$.

Beweis: (i) Man sieht leicht, daß jede Regel korrekt ist, d.h. von gültigen zu gültigen Sequenzen führt, und offenbar sind alle Axiome gültig.

(ii) Sei $\Theta \vartriangleright D$ eine unbeweisbare Sequenz. Sei Σ die endliche Menge der in Θ oder D vorkommenden Kategorien, und $\mathcal{M}' = (2^{\Sigma^+}, \cdot, \backslash, /)$, wobei $\cdot$ die Anhebung der Verkettung von Kategorienfolgen auf Mengen ist. Für Basiskategorien B sei

$$[\![B]\!] := \{\, \Delta \mid \Delta \vdash B \,\},$$

wobei Δ über nichtleere Folgen von Kategorien über Σ läuft. Dann ist

$$[\![C]\!] = \{\, \Gamma \mid \Gamma \vdash C \,\} \tag{53}$$

für jede Kategorie C über Σ gilt. Ist nämlich $C = A/B$, so ist nach Definition von $[\![A/B]\!]$ und der Induktionsvoraussetzung

$$[\![A/B]\!] = \{\, \Gamma \mid \Gamma, \Delta \vdash A \text{ für jedes } \Delta \text{ mit } \Delta \vdash B \,\}.$$

Für jedes $\Gamma \in [\![A/B]\!]$ ist wegen $B \vdash B$ und der Regel $(/\ R)$ auch $\Gamma \vdash A/B$. Für die Umkehrung sei $\Gamma \vdash A/B$ und $\Delta \vdash B$. Die Ableitung

$$\dfrac{\Gamma \rhd A/B \qquad \dfrac{\Delta \rhd B \qquad A \rhd A}{A/B, \Delta \rhd A}\,(/\ L)}{\Gamma, \Delta \rhd A}\,(\mathit{Schnitt})$$

zeigt, daß $\Gamma \in [\![A/B]\!]$. Analog geht die Behauptung für $B\backslash A$. Aus (53) folgt, daß $\mathcal{M} = (2^{\Sigma^+}, \backslash, /, [\![B_i]\!])_{i \in I\!N}$ ein Sprachenmodell ist, in dem $\Theta \rhd D$ nicht gilt. □

Das im Beweis konstruierte 'kanonische Modell' $\mathcal{M}$ ist aber kein Sprachenmodell für die Kategorien mit $\cdot$, da $[\![A \cdot B]\!] = [\![A]\!] \cdot [\![B]\!]$ nicht gilt: zwar ist $(A \cdot B)/B, B \vdash A \cdot B$, aber $(A \cdot B)/B \rhd A$ ist nicht beweisbar. Erst 1993 zeigte M. Pentus[17], daß der Lambek'sche Kalkül auch für Sequenzen *mit* $\cdot$ bezüglich der Sprachenmodelle vollständig ist.

Folgerung 10.14 Für zwei Kategorien A und B sind folgende Aussagen äquivalent:

(i) $[\![A]\!] = [\![B]\!]$ gilt in jedem Standardmodell $\mathcal{M}$.

(ii) Im Lambek'schen Kalkül sind $A \rhd B$ und $B \rhd A$ beweisbar.

Adäquatheit von Lambek-Grammatiken

Wir zeigen nun einen gewissen Nachteil des Lambek'schen Kalküls, wenn man damit –durch eine Kategorialgrammatik– eine Sprache definieren will: die durch eine Kategorialgrammatik G definierten Sprachen $A^G \subseteq \Sigma^+$ bilden kein Standardmodell des Kalküls.

Definition 10.15 Ein Standardmodell $\mathcal{M} \subseteq (2^{\Sigma^+}, \cdot, \backslash, /, [\![B_i]\!])_{i \in I\!N}$ ist ein Modell des Lexikons $T : \Sigma \to 2^{\mathrm{Cat}}$, kurz: $\mathcal{M} \models T$, falls für jede Kategorie $A \in \mathrm{Cat}$

$$A^T := \{\, a \in \Sigma \mid A \in T(a) \,\} \;\subseteq\; [\![A]\!].$$

Beachte, daß $A^G = \bigcup\{ A_1^T \cdots A_n^T \mid A_1, \dots, A_n \vdash A \}$ und daher $A^T \subseteq A^G$.

Proposition 10.16 Ist $G = (\Sigma, T, S)$ eine Lambek'sche Kategorialgrammatik, so ist

$$A^G = \bigcap\{ [\![A]\!]^{\mathcal{M}} \mid \mathcal{M} \models T \}$$

Beweis: Das folgt[11] aus dem Vollständigkeitssatz von Buszkowski und Pentus, wegen

$$T \vdash w : A \iff \forall \mathcal{M} \models T\ \mathcal{M} \models w : A \iff w \in \bigcap\{ [\![A]\!]^{\mathcal{M}} \mid \mathcal{M} \models T \}.$$

□

Da die Divisionen für Wortmengen durch die Wortverkettung festgelegt sind, legt jede Kategorialgrammatik G durch die Basiskategorien B_i^G ein Sprachenmodell

$$\mathcal{M}^G := (2^{\Sigma^+}, \cdot, \backslash, /, B_i^G)_{i \in I\!N}$$

fest. Wegen $A^T \subseteq A^G$ und (10.16) folgt direkt:

$$\mathcal{M}^G \models T \text{ genau dann, wenn } A^G \subseteq [\![A]\!] \text{ für alle } A \in \textsc{Cat}. \tag{54}$$

Nach (10.16) sind die durch G definierten Sprachen A^G der gemeinsame Anteil der Sprachen $[\![A]\!]$ in den verschiedenen Standardinterpretationen des Lexikons von G. Der folgende Satz zeigt, daß sie aber *nicht* die Werte $[\![A]\!]$ eines einzelnen Standardmodells von T sind, für das nur $\mathcal{M}^G$ in Frage käme.

Satz 10.17 (Buszkowski 1982) Sei $G = (\Sigma, T, S)$ eine Lambek-Grammatik. Es gibt Kategorien $C \in \textsc{Cat}$ mit $C^G \neq [\![C]\!]^{\mathcal{M}^G}$, wobei die in C vorkommenden Basiskategorien auch in T auftreten.

Beweis: Wegen $[\![A]\!]/[\![B]\!] = [\![A/B]\!]$ genügt es, Kategorien A, B mit $A^G/B^G \not\subseteq (A/B)^G$ anzugeben. (Da $A/B, B \vdash A$, ist $(A/B)^G \cdot B^G \subseteq A^G$, also stets $(A/B)^G \subseteq A^G/B^G$.) Sei p eine in T auftretende Basiskategorie, und $p^0 := p$, $p^{n+1} := p/p^n$. Wir zeigen, daß für hinreichend großes n dann

$$(p^{n+1})^G = (p^n)^G = \emptyset \tag{55}$$

und daher

$$p^G/(p^n)^G = p^G/\emptyset = \Sigma^+ \not\subseteq \emptyset = (p^{n+1})^G = (p/p^n)^G$$

[11] Diese Behauptung ist *falsch*: es folgt nur die Richtung $\Longrightarrow$. Es ist anscheinend nicht bekannt, ob die Richtung $\Longleftarrow$ gilt.

ist. Die Behauptung (55) ist gleichbedeutend damit, daß bei großem n

$$A_1, \ldots, A_m \not\vdash p^n \quad \text{und} \quad A_1, \ldots, A_m \not\vdash p/p^n \tag{56}$$

für alle $A_1, \ldots, A_m \in \bigcup\{\, T(a) \mid a \in \Sigma \,\}$ und $m \geq 1$.

Wir zeigen nun allgemeiner, daß die p^n nicht aus Annahmen mit beschränkter Komplexität herleitbar sind. Definiere die Ordnung $o(C)$ einer Kategorie durch

$$\begin{aligned} o(C) &:= \{\, 0, \quad \text{falls } C \text{ eine Basiskategorie ist} \\ o(A/B) = o(B\backslash A) &:= \begin{cases} o(A), & \text{falls } o(B) < o(A), \\ o(B)+1, & \text{sonst.} \end{cases} \end{aligned}$$

Beachte, daß $o(p^n) = n$ für alle $n \geq 0$. Für $C \in \textsc{Cat}_n := \{\, C \in \textsc{Cat} \mid o(c) \leq n \,\}$ sei der n-Rang $r_n(C)$ definiert durch

$$\begin{aligned} r_n(C) &:= 0, \quad \text{falls } C \text{ eine Basiskategorie ist,} \\ r_n(B\backslash A) &:= r_n(A), \\ r_n(A/B) &:= \begin{cases} r_n(A)+1, & \text{falls } r_n(A) > 0, \\ 0, & \text{falls } r_n(A) = 0 \text{ und } \Gamma \vdash B \text{ für ein } \Gamma \in \textsc{Cat}^+_{n-2}, \\ 1, & \text{sonst.} \end{cases} \end{aligned}$$

wobei $\textsc{Cat}_{-1} := \emptyset$. Da für großes n jedes $T(a) \subseteq \textsc{Cat}_n$ ist, ergibt sich (56) aus folgender

Beh. Für alle $A_1, \ldots, A_m, A, B \in \textsc{Cat}_n$ gelten:

$$n \geq 1,\ A_1, \ldots, A_m, B \vdash A \quad \Rightarrow \quad r_n(B) \leq r_n(A) \tag{57}$$

$$A_1, \ldots, A_m, B \not\vdash p^{n+1} \quad \text{und} \quad r_{n+1}(p^{n+1}) = 1. \tag{58}$$

Beweis der zweiten Aussage: $n = 0$: Nach den Definitionen ist $r_1(p/p) = 1$. Wäre $\Gamma := A_1, \ldots, A_m, B$ eine nichtleere Folge von Basiskategorien mit $\Gamma \vdash p/p$, so wäre $\Gamma, p \rhd p$ herleitbar, aber diese Sequenz ist nicht allgemeingültig. Daher ist $\Gamma \not\vdash p/p$. Gilt für ein n die Aussage schon, so ist $r_{n+2}(p/p^{n+1}) = 1$. Wäre $\Gamma \vdash p/p^{n+1}$, so auch $\Gamma, p^{n+1} \vdash p$, und mit der ersten Aussage daher $r_{n+1}(p^{n+1}) \leq r_{n+1}(p) = 0$, im Widerspruch zur Annahme $r_{n+1}(p^{n+1}) = 1$.

Es bleibt die erste Aussage zu beweisen. Dazu definieren wir Kategorien C/Γ durch

$$C/(A_1, \ldots, A_n) := \begin{cases} C, & \text{falls } n = 0 \text{ ist,} \\ (C/A_n)/(A_1, \ldots, A_{n-1}), & \text{sonst.} \end{cases}$$

Für $n \geq 1$, $A, B \in \textsc{Cat}_n$, $C \in \textsc{Cat}_{n-1}, \Gamma \in \textsc{Cat}^+_{n-1}$ und $\Delta \in \textsc{Cat}^*_{n-1}$ sind folgende Eigenschaften leicht zu überprüfen:

$$r_n(C) = 0, \tag{59}$$

$$r_n(A/\Gamma) \leq r_n(B/\Gamma), \qquad \text{falls } r_n(A) \leq r_n(B) \tag{60}$$

$$r_n(A) \leq r_n(A/\Gamma), \tag{61}$$

$$r_n((C\backslash A)/\Delta) = r_n(A/\Delta), \tag{62}$$

$$r_n(A/C) \leq r_n(A/\Gamma), \qquad \text{falls } \Gamma \vdash C. \tag{63}$$

Damit zeigen wir folgende Verstärkung von (57):

$$n \geq 1, \Gamma \in \textsc{Cat}_n^*,\ A, B \in \textsc{Cat}_n,\ \Delta \in \textsc{Cat}_{n-1}^*,\ \Gamma, B, \Delta \vdash A \Longrightarrow r_n(B) \leq r_n(A/\Delta) \quad (64)$$

Beweis: durch Induktion über die Länge der Herleitung von $\Gamma, B, \Delta \vdash A$. Falls keine Regel benutzt wurde, ist die Sequenz $\Gamma, B, \Delta \rhd A$ ein Axiom, also $B = A$ und Γ, Δ leer, und die Behauptung $r_n(B) \leq r_n(A)$ gilt offensichtlich. Sonst betrachten wir die zuletzt verwendete Regel; wegen (10.8) ist das o.B.d.A. nicht die Schnittregel.

a) Die Herleitung endet mit $B = B_1 \backslash B_2$ in

$$\frac{\Gamma_1 \rhd B_1 \qquad \Gamma_2, B_2, \Delta \rhd A}{\Gamma_2, \Gamma_1, B_1 \backslash B_2, \Delta \rhd A} (\backslash\ L)$$

Dann ist nach der Induktionsannahme auch $r_n(B_1 \backslash B_2) = r_n(B_2) \leq r_n(A/\Delta)$.

b) Die Herleitung endet in

$$\frac{\Gamma_1 \rhd B_1 \qquad \Gamma_2, B_2, \Gamma_3, B, \Delta, \rhd A}{\Gamma_2, \Gamma_1, B_1 \backslash B_2, \Gamma_3, B, \Delta \rhd A} (\backslash\ L)$$

Die Behauptung gilt nach der Induktionsannahme über die rechte Obersequenz.

c) Die Herleitung endet in

$$\frac{\Gamma_2, B, \Delta_1 \rhd B_1, \qquad \Gamma_1, B_2, \Delta_2 \rhd A}{\Gamma_1, \Gamma_2, B, \Delta_1, B_1 \backslash B_2, \Delta_2 \rhd A} (\backslash\ L)$$

Nach (59) ist $r_n(B_1) = 0 \leq r_n(A)$, und daher mit der Induktionsannahme für die linke Obersequenz nach (60) und (61) auch

$$r_n(B) \leq r_n(B_1/\Delta_1) \leq r_n(A/\Delta_1) \leq r_n(A/(\Delta_1, B_1 \backslash B_2, \Delta_2)).$$

d) Die Herleitung endet in

$$\frac{\Delta_2 \rhd B_1, \qquad \Gamma, B, \Delta_1, B_2, \Delta_3 \rhd A}{\Gamma, B, \Delta_1, \Delta_2, B_1 \backslash B_2, \Delta_3 \rhd A} (\backslash\ L).$$

Dann ist auch $\Delta_2, B_1 \backslash B_2 \rhd B_2$ herleitbar, und daher sind $r_n(A/B_2) \leq r_n(A/(\Delta_2, B_1 \backslash B_2))$ und $r_n(A/(B_2, \Delta_3)) \leq r_n(A/(\Delta_2, B_1 \backslash B_2, \Delta_3))$ nach (63); da $\Delta_1 \in \textsc{Cat}_{n-1}^*$ ist, folgt weiter $r_n(A/(\Delta_1, B_2, \Delta_3)) \leq r_n(A/(\Delta_1, \Delta_2, B_1 \backslash B_2, \Delta_3))$ mit $(A/(D, \Delta)) = (A/\Delta)/D$. Die Induktionsannahme über die rechte Obersequenz liefert die noch fehlende Ungleichung $r_n(B) \leq r_n(A/(\Delta_1, B_2, \Delta_3))$.

e) - i) Die Herleitung endet in einer Anwendung der Regel $(/\ L)$. Diese Fälle gehen analog zu den Fällen a) - d) und bleiben dem Leser überlassen.

j) Die Herleitung endet mit $A = A_1\backslash A_2$ in

$$\frac{A_1, \Gamma, B, \Delta \ \triangleright\ A_2}{\Gamma, B, \Delta \ \triangleright\ A_1\backslash A_2}\ (\backslash\ R)$$

Nach Induktion ist $r_n(B) \leq r_n(A_2/\Delta)$, und wegen $A \in \textsc{Cat}_n$ ist $A_1 \in \textsc{Cat}_{n-1}$. Daher ist mit (62) $r_n(A_2/\Delta) = r_n((A_1\backslash A_2)/\Delta)$, und die Behauptung gilt.

k) Die Herleitung endet mit $A = A_2/A_1$ in

$$\frac{\Gamma, B, \Delta, A_1 \ \triangleright\ A_2}{\Gamma, B, \Delta \ \triangleright\ A_2/A_1}\ (/\ R)$$

Nach Induktion ist $r_n(B) \leq r_n(A_2/(\Delta, A_1)) = r_n((A_2/A_1)/\Delta)$. Damit ist die Behauptung (für den Kalkül ohne $\cdot$) gezeigt. $\square$

Der Satz zeigte, daß die durch das Lexikon T einer Grammatik $G = (\Sigma, T, S)$ und den Lambek-Kalkül axiomatisierte Theorie in dem Sinne unvollständig ist, daß sie nicht alle wahren Inklusionen zwischen Sprachen A^G und B^G zu beweisen erlaubt. Es ist noch nicht einmal so, daß die durch G gegebenen Bedeutungen A^G eine korrekte Interpretationen der Theorie bilden, d.h. alle beweisbaren Inklusionen erfüllen (siehe Aufgabe 10.2).

Durch den Lambek'schen Kalkül werden nur die in Sprachmodellen *allgemein*gültigen Inklusionen oder Gleichungen zwischen Kategorien erfaßt. In einem bestimmten Sprachmodell gelten i.a. weitere Gleichungen bzw. Inklusionen zwischen Kategorien, die keiner beweisbaren Sequenz entsprechen.

Aufgaben

Aufgabe 10.1 Zeige, daß zu jeder Kategorialgrammatik im Sinne von Ajdukiewicz (d.h. nach Definition 10.5) eine äquivalente Lambek-Kategorialgrammatik konstruiert werden kann.

Hinweis: (a) Für jede Kategorie C gilt $C \vdash C$. (b) Die Kürzungsregel $A \cdot (A\backslash B) \subseteq B$ kann im Lambek-Kalkül durch

$$\cfrac{\cfrac{A \ \triangleright\ A \qquad\qquad B \ \triangleright\ B}{A, A\backslash B \ \triangleright\ B}\ (\backslash\ L)}{A \cdot (A\backslash B) \ \triangleright\ B}\ (\cdot\ L)$$

bewiesen werden, wobei man (a) für die Annahmen benutzt.

Aufgabe 10.2 (Buszkowski 1982) Sei $G = (\{a, b\}, T, S)$ mit $T(a) = \{p/(q/q)\}$ und $T(b) = \{q/q\}$. Zeige, daß $[\![(p/(q/q))]\!] = \emptyset$ in $\mathcal{M}^G$ und daher $\mathcal{M}^G \not\models T$.

11 Rekursiv aufzählbare Sprachen

Bisher haben wir Sprachen durch simultane induktive Definition der grammatischen Kategorien betrachtet. Diese Art der Definition ist im wesentlichen ausreichend für logische Formelsprachen und Programmiersprachen. Sie hat einen sehr konstruktiven Charakter, da größere Ausdrücke aus kleineren zusammengesetzt werden.

Eine natürliche Klasse von Sprachdefinitionen, die diese induktiven Definitionen umfaßt, erhält man aus Eigenschaften von Zeichenreihen, die sich durch logische Formeln ausdrücken lassen. Jede logische Formel $\varphi(x)$ definiert eine Sprache

$$L(\varphi) := \{\, w \in \Sigma^* \mid \mathcal{A} \models \varphi(w) \,\},$$

wobei $\mathcal{A}$ eine unendliche Struktur wie $(\Sigma^*, \cdot, a)_{a\in\Sigma}$ oder eine endliche Struktur wie die von $w \equiv a_1 \cdots a_n$ abhängenden Strukturen

$$(\Sigma^{\leq|w|}, \cdot, a)_{a\in\Sigma} \quad \text{oder} \quad (\{0, 1, \dots, |w|\}, 0, P_a)_{a\in\Sigma} \text{ mit } P_a(i) \iff a \equiv a_i$$

sein kann. Je nach den erlaubten Grundprädikaten (z.B. der Relation $x = y \cdot z$) und logischen Mitteln (den Junktoren und Quantoren in φ) und ihrer Interpretation in $\mathcal{A}$ (z.B. über welchen Bereich die Quantoren rangieren) erhält man unterschiedliche Sprachen.

Wir betrachten im Folgenden eine solche Form der Sprachdefinition, bei der man Sprachen als Lösungsmengen von Wortgleichungen in $(\Sigma^*, \cdot, a)_{a\in\Sigma}$ betrachtet, d.h. wo φ eine Gleichung zwischen Wörtern über $\Sigma \cup \{x\}$ ist.

11.1 Wortprobleme von Thue- und Semi-Thue-Systemen

Eine Äquivalenzrelation $R \subseteq \Sigma^* \times \Sigma^*$ heißt KONGRUENZRELATION auf $(\Sigma^*, \cdot)$, falls sie mit der Verkettung $\cdot$ verträglich ist, d.h. falls für alle $u, v, w \in \Sigma^*$ gilt:

$$(v, w) \in R \implies (uv, uw) \in R, \quad (vu, wu) \in R.$$

Jede Relation $E \subseteq \Sigma^* \times \Sigma^*$ erzeugt eine Kongruenzrelation $=_E$ auf Σ^*, nämlich

$$=_E \; := \bigcap \{\, R \mid E \subseteq R \subseteq \Sigma^* \times \Sigma^*,\ R \text{ ist eine Kongruenzrelation auf } (\Sigma^*, \cdot) \,\},$$

die kleinste Kongruenzrelation auf Σ^*, die E umfaßt. Man kann sie mit den folgenden Regeln auch 'von unten' aufbauen:

$$\text{(E)} \quad \frac{(u, v) \in E}{u =_E v} \qquad\qquad \text{(Kongr)} \quad \frac{v =_E v'}{uvw =_E uv'w}$$

$$\text{(Refl)} \quad v =_E v \qquad \text{(Sym)} \quad \frac{v =_E w}{w =_E v} \qquad \text{(Trans)} \quad \frac{u =_E v, \quad v =_E w}{u =_E w}$$

Ist $E \subseteq \Sigma^* \times \Sigma^*$ endlich, so nennt man $(\Sigma^*, \cdot, E)$ ein THUE-SYSTEM, nach A. Thue[26], der verschiedene Probleme über Zeichenreihen untersucht hat. Das WORTPROBLEM FÜR $(\Sigma^*, \cdot, E)$ besteht darin, zu Wörtern $u, v \in \Sigma^*$ festzustellen, ob $u =_E v$ gilt.

Das allgemeinere Problem der LÖSBARKEIT VON $=_E$-GLEICHUNGEN ÜBER $(\Sigma^*, \cdot, E)$ besteht darin, zu $u, v \in (\Sigma \cup \{x_1, \ldots, x_m\})^*$ festzustellen, ob es $\vec{w} = w_1, \ldots, w_m \in \Sigma^*$ mit $u[\vec{w}/\vec{x}] =_E v[\vec{w}/\vec{x}]$ gibt. Die LÖSUNGSMENGE

$$L(u =_E v) \;:=\; \{\, \vec{w} \in (\Sigma^*)^m \mid u[\vec{w}/\vec{x}] =_E v[\vec{w}/\vec{x}] \,\}$$

einer solchen Gleichung ist eine mit endlichen Mitteln, nämlich u, v und E, definierte m-stellige Relation auf Σ^*. Ihre Projektion auf eine Komponente, z.B.

$$L(u =_E v)_1 \;:=\; \{\, w_1 \in \Sigma^* \mid \exists w_2, \ldots, w_m \in \Sigma^* \; u[\vec{w}/\vec{x}] =_E v[\vec{w}/\vec{x}] \,\},$$

ist also eine endlich repräsentierte Sprache über Σ. Man kann daher auch die Beschreibung einer Sprache als (Projektion der) Lösungsmenge einer Wortgleichung als eine Grammatik verstehen. Allerdings ergibt sich die Zugehörigkeit eines Ausdrucks w zur definierten Sprache nicht mehr in naheliegender Weise aus Eigenschaften seiner Teilausdrücke. Es stellt sich heraus, daß es im Allgemeinen überhaupt keine Verfahren gibt, mit denen man feststellen könnte, ob w zu einer Lösung $\vec{w} = w_1, \ldots, w_m$ einer gegeben Wortgleichung $u(\vec{x}) =_E v(\vec{x})$ ergänzt werden kann.

Uns genügt im Folgenden der Spezialfall, wo u nur eine und v keine Variable enthält, also

$$L(u =_E v) \;=\{\, w \in \Sigma^* \mid u[w/x] =_E v \,\}. \tag{65}$$

Wir betrachten auch davon nur eine Abschwächung, bei der man die Symmetrie der Relation $=_E$ nicht fordert:

Definition 11.1 Ein SEMI-THUE-SYSTEM $(\Sigma^*, \cdot, E)$ besteht aus einem endlichen Alphabet Σ und einer endlichen Relation $E \subseteq \Sigma^* \times \Sigma^*$. Man nennt $(v, v') \in E$ eine ERZEUGENDE RELATION. E definiert eine Relation $\Rightarrow_E \subseteq \Sigma^* \times \Sigma^*$ zwischen Wörtern über Σ, durch

$$w_1 \Rightarrow_E w_2 :\Longleftrightarrow \; \exists\, (v, v') \in E\, \exists u_1, u_2 \in \Sigma^* \, (w_1 = u_1 v u_2 \wedge w_2 = u_1 v' u_2).$$

Das WORTPROBLEM für das Semi-Thue-System $(\Sigma^*, \cdot, E)$ besteht darin, zu beliebigen $w_1, w_2 \in \Sigma^*$ festzustellen, ob $w_1 \Rightarrow_E^* w_2$.

Offenbar ist $\Rightarrow_E^*$ die kleinste reflexive transitive Relation oberhalb von E, die mit der Verkettung verträglich ist; und das ist $=_E$, falls E symmetrisch ist. Beachte, daß $w_1 \Rightarrow_E w_2$ eine Anwendung der Regel (E) gefolgt von einer Anwendung der Regel (Kongr) ist.

Durch Semi-Thue-Systeme wird der Prozeß der *Veränderung* von Zeichenreihen nach endlich vielen Regeln hervorgehoben. Die Veränderung erfolgt *lokal*, da nur Ersetzungen von

Teilwörtern erlaubt sind, die einer Regelseite (von beschränkter Länge) entsprechen, und *schrittweise*, weshalb man sich, falls zwei Regeln auf sich überlappende Teilworte anwendbar sind, für eine der Regeln entscheiden muß.

Faßt man eine Zeichenreihe als Speicher einer Maschine auf, dessen Größe variiert, so kann man diese lokale, schrittweise Veränderung des Speichers als einen Berechnungsprozeß ansehen. Hierbei ist die unsymmetrische Form der Semi-Thue-Systeme angemessen. Über diesen Zusammenhang erhält man:

Satz 11.2 Es gibt Semi-Thue-Systeme, deren Wortproblem unentscheidbar ist.

Das werden wir in Abschnitt 11.3 ausführen. Der symmetrischen Form, also den Thue-Systemen, entsprechen *symmetrische* Maschinen, deren Instruktionen auch rückwärts ausgeführt werden können; sie sind schwerer zu programmieren.

Wortprobleme von Thue-Systemen treten in der Mathematik auf, etwa in der Gruppentheorie: Sei Σ eine Menge von 'Erzeugenden', und zu jedem Element a gebe es auch ein Inverses a^{-1}; neben den Gleichungen $\{\, a \cdot a^{-1} = 1 \mid a \in \Sigma \,\}$ enthalte E weitere Gleichungen, die Beziehungen zwischen den Erzeugenden ausdrücken, z.B. $a_1 \cdot a_2 = a_3^{-1} \cdot a_1 \cdot a_2$. (Daraus folgen weitere Gleichungen, etwa $a_1 = a_3^{-1} \cdot a_1$.) P.S. Novikov und W.W. Boone[28] haben Gruppen $(G, \cdot, 1)$ mit erzeugenden Relationen E angegeben, für die das Wortproblem unentscheidbar ist.

11.2 Generative Phrasenstrukturgrammatiken

Der Linguist N.Chomsky hat die schrittweise lokale Veränderung von Zeichenreihen als einen *Erzeugungsprozeß* von sprachlichen Ausdrücken interpretiert. Da die Gesamtheit der grammatisch korrekten Sätze einer Sprache unendlich ist, aber jedes Kind nach kurzer Zeit auch Sätze bildet, die es nie vorher gehört hat, ist es plausibel anzunehmen, daß die Erzeugung von Sätzen durch eine endliche Menge von Regeln gesteuert wird, die sich in hinreichend kurzer Zeit lernen lassen.

In den 'generativen Phrasenstrukturgramatiken' von Chomsky werden Satzschemata als Zeichenreihen mit Variablen erzeugt; die variablenfreien Satzschemata sind dann die Sätze der Sprache.

Definition 11.3 Eine generative PHRASENSTRUKTURGRAMMATIK $G = (N, \Sigma, P, S)$ besteht aus

(i) endlichen Mengen N und Σ, mit $N \cap \Sigma = \emptyset$, wobei Σ das TERMINAL-, N das NONTERMINALALPHABET von G heißt,

(ii) einem ausgezeichneten Nonterminal $S \in N$, dem START- oder SATZSYMBOL, und

(iii) einer endlichen Menge $P \subseteq (V^* \times V^*) - (\Sigma^* \times V^*) = V^+ \times V^*$, den PHRASENSTRUKTURREGELN, mit $V := N \cup \Sigma$.

Jedes $\alpha \in V^*$ heißt Phrasen- bzw. SATZFORM. Statt $(\alpha, \beta) \in P$ schreibt man $\alpha \to \beta$. Für Phrasen $\varrho, \psi \in V^*$ sagen wir, ψ IST AUS ϱ MIT G IN EINEM SCHRITT ERZEUGBAR, $\varrho \Longrightarrow_G \psi$, falls es eine Regel $\alpha \to \beta \in P$ und $\varrho_1, \varrho_2 \in V^*$ gibt, so daß

$$\varrho = \varrho_1 \alpha \varrho_2 \quad \text{und} \quad \psi = \varrho_1 \beta \varrho_2.$$

Mit $\overset{*}{\Longrightarrow}_G$ und $\overset{n}{\Longrightarrow}_G$ werden die reflexive transitive Hülle und die n-fache Potenz von $\Longrightarrow_G$ bezeichnet. DIE VON G ERZEUGTE SPRACHE ist

$$L(G) := \{w \in \Sigma^* \mid S \overset{*}{\Longrightarrow}_G w\}.$$

Chomsky hat eine Einteilung der Phrasenstrukturgrammatiken nach der Form ihrer Regeln eingeführt:

Definition 11.4 Eine Grammatik $G = (N, \Sigma, P, S)$ ist vom TYP 1 oder KONTEXTSENSITIV, falls jede Regel aus P entweder von der Form

$$\alpha A \gamma \to \alpha \beta \gamma \;\text{ mit }\; A \in N,\; \beta \in (\Sigma \cup N)^* - \{\varepsilon\}$$

oder die Regel $S \to \varepsilon$ ist und in diesem Fall S in keiner anderen Regel rechts auftritt. G heißt KONTEXTFREI oder vom TYP 2, falls jede Regel aus P von der Form

$$A \to \alpha \;\text{ mit }\; A \in N,\; \alpha \in (N \cup \Sigma)^*$$

ist. G heißt EINSEITIG LINEAR oder vom TYP 3, falls jede Regel von der Form

$$A \to wB \;\text{ oder }\; A \to w \;\text{ mit }\; A, B \in N,\; w \in \Sigma^*$$

ist (dann heißt G rechtslinear) oder falls jede Regel von der Form

$$A \to Bw \;\text{ oder }\; A \to w \;\text{ mit }\; A, B \in N,\; w \in \Sigma^*$$

ist (dann heißt G linksinear). G heißt LINEAR, wenn G kontextfrei ist und jede Regel aus P von der Form

$$A \to w_1 B w_2 \;\text{ oder }\; A \to w \quad \text{mit } A, B \in N,\; w_1, w_2 \in \Sigma^*$$

ist. Ist die Form der Regeln nicht eingeschränkt, so heißt G vom TYP 0.

Mit Typ-0-Grammatiken kann man regelrecht rechnen, wie das folgende Beispiel zeigt, bei dem Symbole an passende Stellen eines Wortes 'geschoben' und dort 'umgewandelt' werden:

Beispiel 11.5 Sei $G = (N, \Sigma, P, S)$ mit $N = \{S, A_0, A_1, A_2, B_0, B_1\}$, $\Sigma = \{0, 1\}$ und

$$P = \left\{ \begin{array}{lllllllll} S & \to & A_0A_1A_2, & A_0A_1 & \to & \varepsilon, & A_0A_1 & \to & iA_0B_i, \\ jA_1 & \to & A_1j, & B_ij & \to & jB_i, & B_iA_2 & \to & A_1iA_2, \\ A_2 & \to & \varepsilon & & & & & & \end{array} \middle| \; i, j \in \{0,1\} \right\}$$

Dann ist G eine Typ-0-Grammatik mit $L(G) = \{\, ww \mid w \in \{0,1\}^* \,\}$.

Beweisidee: Für $\{\, ww \mid w \in \{0,1\}^* \,\} \subseteq L(G)$ beachte, daß

$$S \overset{*}{\Longrightarrow} A_0A_1A_2,$$

und daß für jedes $w \in \Sigma^*,\ i \in \Sigma$

$$\begin{array}{rcl} wA_0A_1wA_2 & \overset{*}{\Longrightarrow} & wiA_0B_iwA_2 \\ & \overset{*}{\Longrightarrow} & wiA_0wB_iA_2 \\ & \overset{*}{\Longrightarrow} & wiA_0wA_1iA_2 \\ & \overset{*}{\Longrightarrow} & wiA_0A_1wiA_2. \end{array}$$

Daher ist $ww \in L(G)$ für jedes $w \in \Sigma^*$. Für die Umkehrung muß man sich überlegen, daß jede anders verlaufende Ableitung $S \overset{*}{\Longrightarrow} \alpha$ ein Nonterminal in α hat, das man nicht mehr zu einem Terminalwort reduzieren kann.

Satz 11.6 (Chomsky-Hierarchie) Für $i \leq 3$ sei

$$\mathcal{L}_i(\Sigma) := \{\, L(G) \subseteq \Sigma^* \mid G = (N, \Sigma, P, S) \text{ ist eine Grammatik vom Typ } i \,\}.$$

Falls Σ mindestens zwei Symbole enthält, sind die Sprachklassen $\mathcal{L}_i$ echte Erweiterungen von einander:

$$\mathcal{L}_3(\Sigma) \subset \mathcal{L}_2(\Sigma) \subset \mathcal{L}_1(\Sigma) \subset \mathcal{L}_0(\Sigma).$$

In früheren Abschnitten haben wir schon gesehen, daß $\mathcal{L}_3 = \mathit{Reg} \subset \mathit{Lin} \subset \mathit{CF} = \mathcal{L}_2$ ist.

Kontextsensitive Grammatiken

Bei der Definition kontextfreier Grammatiken haben wir beliebige kontextfreie ε-Regeln $A \to \varepsilon$ zugelassen, bei kontextsensitiven aber nur $S \to \varepsilon$. Daher ist es nicht ganz offensichtlich, daß gilt:

Lemma 11.7 Jede kontextfreie Sprache über Σ ist auch eine kontextsensitive Sprache über Σ: $\mathcal{L}_2(\Sigma) \subseteq \mathcal{L}_1(\Sigma)$.

Beweis: Nach Korollar 6.28 gibt es zu jeder kontextfreien Sprache eine kontextfreie Grammatik, die der Bedingung an ε-Regeln aus der Definition kontextsensitiver Grammatiken genügt. $\square$

Eine Phrasenstrukturregel $\alpha \to \beta$ ist VERKÜRZEND, falls $|\alpha| > |\beta|$, d.h. falls die rechte Seite weniger Symbole als die linke enthält. Außer $S \to \varepsilon$ ist keine Regel einer kontextsensitiven Grammatik verkürzend. Umgekehrt gilt:

Proposition 11.8 Zu jeder Grammatik G ohne verkürzende Regeln gibt es eine äquivalente kontextsensitive Grammatik.

Beweis: Sei $A_1 \cdots A_n \to B_1 \cdots B_{n+k}$ eine nicht verkürzende Regel. Mit neuen Nonterminalen $A_1', \ldots, A_n'$ ersetzen wir sie durch die folgenden kontextsensitiven Regeln:

$$\begin{array}{rcl@{\qquad}rcl}
A_1A_2\cdots A_n & \to & A_1'A_2\cdots A_n & A_1'A_2'\cdots A_n' & \to & B_1A_2'\cdots A_n' \\
A_1'A_2A_3\cdots A_n & \to & A_1'A_2'A_3\cdots A_n & B_1A_2'A_3'\cdots A_n' & \to & B_1B_2A_3'\cdots A_n' \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
A_1'\cdots A_{n-1}'A_n & \to & A_1'\cdots A_n' & B_1\cdots B_{n-1}A_n' & \to & B_1\cdots B_{n-1}B_n\cdots B_{n+k}
\end{array}$$

Die neuen Nonterminale $A_1', \ldots, A_n'$ hängen von der Regel $A_1 \cdots A_n \to B_1 \cdots B_{n+k}$ ab. Sie können daher nur in einer (von anderen Regelanwendungen unterbrochenen) Folge von Anwendungen der neuen Regeln verwendet und wieder beseitigt werden, die eine Anwendung von $A_1 \cdots A_n \to B_1 \cdots B_{n+k}$ simuliert. $\square$

Beispiel 11.9 $L = \{a^nb^nc^n \mid n > 0\}$ ist eine kontextsensitive, nicht kontextfreie Sprache.

Beweis: Nach 8.10 ist L nicht kontextfrei. Aber L wird von der Grammatik G mit den Regeln

$$\begin{array}{rcl@{\qquad}rcl}
S & \to & aSA, & S & \to & aB, \\
BA & \to & bBC, & B & \to & bc, \\
CA & \to & AC, & C & \to & c
\end{array}$$

erzeugt. Da G keine verkürzende Regel enthält, gibt es nach 11.8 eine zu G äquivalente kontextsensitive Grammatik. Durch Induktion sieht man, daß mit den letzten vier Regeln $BA^n \overset{*}{\underset{G}{\Longrightarrow}} b^{n+1}c^{n+1}$ für alle $n \geq 0$, und daher

$$S \overset{n}{\underset{G}{\Longrightarrow}} a^nSA^n \Rightarrow a^naBA^n \overset{*}{\underset{G}{\Longrightarrow}} a^{n+1}b^{n+1}c^{n+1}$$

für alle $n \geq 0$. Also ist $L \subseteq L(G)$. Für die Umkehrung sei $S \overset{*}{\underset{G}{\Longrightarrow}} w \in \Sigma^*$. Ist $S \overset{*}{\underset{G}{\Longrightarrow}} \alpha \overset{*}{\underset{G}{\Longrightarrow}} w$ und α die erste Satzform der Ableitung, in der S nicht vorkommt, so ist $\alpha = a^{n+1}BA^n$ für ein $n \geq 0$. Nun kann man A nur durch die dritte Regel beseitigen, und dann C mit der fünften ganz nach rechts schieben und dort zu c umwandeln. Falls man C vorher durch c ersetzt, bleiben A's übrig. Schließlich ist das letzte B durch bc zu ersetzen. $\square$

Damit ist $\mathcal{L}_2(\Sigma) \neq \mathcal{L}_1(\Sigma)$ für $|\Sigma| \geq 3$; es gilt mit $L' = \{a^nb^na^n \mid n \geq 0\}$ auch für $|\Sigma| \geq 2$.

11.3 Rekursive Kellerautomaten und Turingmaschinen

Mit den generativen Phrasenstrukturgrammatiken hat man eine sehr weite Klasse von Sprachdefinitionen. Für die einfachen Teilklassen der einseitig linearen und der kontextfreien Grammatiken können wir feststellen, ob ein gegebenens Wort w zu der durch die Grammatik G definierten Sprache gehört. Für einseitig lineare Grammatiken geht das mit endlichen Automaten, für kontextfreie Grammatiken mit Kellerautomaten.

Wir wollen nun das gleiche Problem für allgemeine Phrasenstrukturgrammatiken betrachten. Gesucht ist also zu jeder Phrasenstrukturgrammatik G ein Automat M, der die von G erzeugte Sprache erkennt. Nach 11.7 und Beispiel 11.9 braucht man einen Automatentyp, der *mehr* erkennen kann als ein Kellerautomat.

Unter einem (abstrakten) Kellerautomaten $\mathcal{M} = (\Gamma, \Sigma, M, S)$ hatten wir einen Kellerspeicher Γ^* mit einer regulären Menge M von Befehlsfolgen über

$$\Sigma_\Gamma := \{\, push_A, pop_A, read_a \mid A \in \Gamma, a \in \Sigma \,\}$$

verstanden, wobei $\vdash_M$ die durch $\vdash_{\overline{push_A}}$, $\vdash_{\overline{pop_A}}$ und $\vdash_{\overline{read_a}}$ bestimmte Übergangsrelation auf der Menge $\Gamma^* \times \Sigma^*$ der Konfigurationen von $\mathcal{M}$ ist.

Wir hatten in Kapitel 7 gesehen, daß Kellerautomaten, d.h. iterative sequentielle Programme $r \in (\Sigma_\Gamma)^*$ über einem Kellerspeicher Γ^*, nichts anderes sind als rekursive endliche Automaten, d.h. rekursive sequentielle Programme über einem endlichen Speicher, aber mehr als endliche Automaten leisten. Es liegt nahe zu vermuten, daß *rekursive* Programme über einem Kellerspeicher mehr vermögen als iterative. Daher betrachten wir *rekursive* Kellerautomaten, also endliche Systeme von Kellerautomaten, die sich gegenseitig aufrufen können. Wir überlassen diese 'konkrete' Definition dem Leser (vgl. Aufgabe 11.1) und geben hier eine abstraktere Fassung an:

Definition 11.10 Ein (abstrakter) REKURSIVER KELLERAUTOMAT $\mathcal{M} = (\Gamma, \Sigma, M, S)$ besteht aus einem endlichen Eingabealphabet Σ, einem endlichen Kelleralphabet Γ, einem Anfangsinhalt $S \in \Gamma^*$ des Kellers Γ^* und einer *kontextfreien* Menge

$$M \subseteq \Sigma_\Gamma^* := \{\, read_a, push_A, pop_A \mid a \in \Sigma, A \in \Gamma \,\}^*$$

von Befehlsfolgen über Σ_Γ. Die VON $\mathcal{M}$ ERKANNTE SPRACHE ist

$$L(\mathcal{M}) := \{\, w \in \Sigma^* \mid (S, w) \vdash_M (\varepsilon, \varepsilon) \,\}.$$

Dies entspricht der Verallgemeinerung von endlichen zu rekursiven endlichen Automaten über Σ: während die Zulässigkeit der Befehlsfolgen $\alpha \in M$ bei einem Kellerautomaten $\mathcal{M} = (\Gamma, \Sigma, M, S)$ durch einen endlichen Automaten $\mathcal{A} = (Q, \vdash_b, q_0, F)_{b \in \Sigma_\Gamma}$ kontrolliert wird, d.h. $M = L(\mathcal{A})$ ist, wird sie bei einem rekursiven Kellerautomaten durch einen rekursiven endlichen Automaten

$$\mathcal{A}' = (Q', \vdash_b, \vdash_{call}, \vdash_{return}, q_0', F')_{b \in \Sigma_\Gamma}$$

(oder einen einfachen Kellerautomaten über Σ_Γ) kontrolliert; nach Satz 7.8 sind solche Mengen $M = L(\mathcal{A}')$ gerade die kontextfreien Mengen $M \subseteq \Sigma_\Gamma^*$.

Nach Satz 6.14 können wir gleichwertig sagen, daß M im ersten Fall die operationale Bedeutung eines iterativen, im zweiten Fall die eines rekursiven Programms r über Σ_Γ ist, d.h. es ist $M = L(r)$ für einen regulären bzw. einen μ-regulären Ausdruck r über Σ_Γ.

Einen rekursiven Kellerautomaten kann man als einen einfachen Kellerautomaten mit einem weiteren Keller zur Verwaltung der Rücksprungadressen der Rekursionsaufrufe darstellen:

Satz 11.11 Zu jedem rekursiven Kellerautomaten $\mathcal{M}_1 = (\Gamma_1, \Sigma, M_1, S_1)$ mit einem Keller Γ_1^* gibt es einen Kellerautomaten $\mathcal{M} = (\Gamma_1, \Gamma_2, \Sigma, M, S_1, S_2)$ mit zwei Kellern, Γ_1^* und Γ_2^*, so daß $L(\mathcal{M}) = L(\mathcal{M}_1)$.

Beweis: Da $M_1 \subseteq \Sigma_{\Gamma_1}^*$ kontextfrei ist, gibt es nach Satz 7.8 einen Kellerautomaten $\mathcal{M}_2 = (\Sigma_{\Gamma_1}, \Gamma_2, M_2, S_2)$ mit

$$M_1 = L(\mathcal{M}_2) = \{\, \beta \in \Sigma_{\Gamma_1}^* \mid (S_2, \beta) \vdash_{M_2} (\varepsilon, \varepsilon) \,\}.$$

Dabei ist

$$M_2 \subseteq ((\Sigma_{\Gamma_1})_{\Gamma_2})^* = \{\, read_b, push_B, pop_B \mid b \in \Sigma_{\Gamma_1}, B \in \Gamma_2 \,\}^*$$

eine reguläre Menge von Befehlsfolgen. Sei $h : (\Sigma_{\Gamma_1})_{\Gamma_2} \to \Sigma_{(\Gamma_1 \cup \Gamma_2)}$ die durch

$$\begin{array}{ll} h(read_b) := b, & b \in \Sigma_{\Gamma_1} = \{\, read_a, push_A, pop_A \mid a \in \Sigma, A \in \Gamma_1 \,\} \\ h(push_B) := push_B, & B \in \Gamma_2, \\ h(pop_B) := pop_B, & B \in \Gamma_2 \end{array}$$

definierte Ersetzung. Nach Proposition 4.24 ist mit M_2 auch $M := h(M_2) \subseteq (\Sigma_{(\Gamma_1 \cup \Gamma_2)})^*$ eine reguläre Menge. Sei

$$\mathcal{M} := (\Gamma_1, \Gamma_2, \Sigma, M, S_1, S_2).$$

Dann ist $\mathcal{M}$ ein Kellerautomat mit Eingabealphabet Σ, zwei Kelleralphabeten Γ_1, Γ_2 und Anfangsinhalten $S_1 \in \Gamma_1^*$ und $S_2 \in \Gamma_2^*$: für $a \in \Sigma$ und $C \in \Gamma_1 \cup \Gamma_2$ interpretieren wir die Übergangsrelationen $\vdash_{read_a}$, $\vdash_{push_C}$ und $\vdash_{pop_C}$ auf der Menge $\Gamma_1^* \times \Gamma_2^* \times \Sigma^*$ aller Konfigurationen von $\mathcal{M}$ gerade so, daß für alle $A \in \Gamma_1$, $B \in \Gamma_2$ und alle $\gamma_i \in \Gamma_i^*$ und $w \in \Sigma^*$ gilt:

$$\begin{array}{lll@{\qquad}lll} (\gamma_1, \gamma_2, aw) & \vdash_{read_a} & (\gamma_1, \gamma_2, w), & & & \\ (\gamma_1, \gamma_2, w) & \vdash_{push_A} & (\gamma_1 A, \gamma_2, w), & (\gamma_1, \gamma_2, w) & \vdash_{push_B} & (\gamma_1, \gamma_2 B, w), \\ (\gamma_1 A, \gamma_2, w) & \vdash_{pop_A} & (\gamma_1, \gamma_2, w), & (\gamma_1, \gamma_2 B, w) & \vdash_{pop_B} & (\gamma_1, \gamma_2, w). \end{array}$$

O.B.d.A. ist dabei $\Gamma_1 \cap \Gamma_2 = \emptyset$, so daß bei $push_C$ und pop_C an C abgelesen werden kann, auf welchem Stapel die Veränderung vorzunehmen ist.

Beh.: Sei $\beta \in (\Sigma_{\Gamma_1})^*_{\Gamma_2}$ und $h(\beta)' \in \Sigma^*_{\Gamma_1}$ aus $h(\beta) \in \Sigma^*_{(\Gamma_1 \cup \Gamma_2)}$ durch Weglassen aller $push_B$ und pop_B mit $B \in \Gamma_2$ entstanden. Dann ist

$$(\gamma_1, \gamma_2, w) \vdash_{\overline{h(\beta)}} (\delta_1, \delta_2, \varepsilon) \iff (\gamma_1, w) \vdash_{\overline{h(\beta)'}} (\delta_1, \varepsilon) \;\wedge\; (\gamma_2, h(\beta)') \vdash_{\overline{\beta}} (\delta_2, \varepsilon) \tag{66}$$

für alle $\gamma_1, \delta_1 \in \Gamma_1^*$, $\gamma_2, \delta_2 \in \Gamma_2^*$ und $w \in \Sigma^*$. Daraus folgt die Behauptung $L(\mathcal{M}_1) = L(\mathcal{M}')$, denn für jedes $w \in \Sigma^*$ ist

$$\begin{aligned}
w \in L(\mathcal{M}_1) \quad &\iff \quad (S_1, w) \vdash_{M_1} (\varepsilon, \varepsilon) \\
&\iff \quad \exists \alpha \in M_1 \; (S_1, w) \vdash_{\alpha} (\varepsilon, \varepsilon) \\
&\iff \quad \exists \beta \in M_2 \, \exists \alpha \in \Sigma^*_{\Gamma_1} \; [(S_1, w) \vdash_{\alpha} (\varepsilon, \varepsilon) \;\wedge\; (S_2, \alpha) \vdash_{\beta} (\varepsilon, \varepsilon)] \\
&\iff \quad \exists \beta \in M_2 \; [(S_1, w) \vdash_{\overline{h(\beta)'}} (\varepsilon, \varepsilon) \;\wedge\; (S_2, h(\beta)') \vdash_{\overline{\beta}} (\varepsilon, \varepsilon)] \\
&\overset{(66)}{\iff} \quad \exists \beta \in M_2 \; (S_1, S_2, w) \vdash_{\overline{h(\beta)}} (\varepsilon, \varepsilon, \varepsilon) \\
&\iff \quad (S_1, S_2, w) \vdash_{\overline{h(M_2)}} (\varepsilon, \varepsilon, \varepsilon) \\
&\iff \quad w \in L(\mathcal{M}_2).
\end{aligned}$$

Man zeigt (66) durch Induktion über die Form von β.

$\beta \equiv \varepsilon$: Wegen $h(\beta) \equiv \varepsilon \equiv h(\beta)'$ sind beide Seiten von (66) äquivalent zu $\gamma_1 \equiv \delta_1$, $\gamma_2 \equiv \delta_2$, $w \equiv \varepsilon$.

$\beta \equiv read_{push_A}\delta$: Dann ist $h(\beta) = push_A h(\delta)$ und $h(\beta)' = push_A h(\delta)'$ wegen $A \in \Gamma_1$. Also ist

$$\begin{aligned}
&(\gamma_1, \gamma_2, w) \vdash_{\overline{h(\beta)}} (\delta_1, \delta_2, \varepsilon) \\
&\iff \quad (\gamma_1, \gamma_2, w) \vdash_{\overline{push_A}} (\gamma_1 A, \gamma_2, w) \vdash_{\overline{h(\delta)}} (\delta_1, \delta_2, \varepsilon) \\
&\iff \quad (\gamma_1 A, w) \vdash_{\overline{h(\delta)'}} (\delta_1, \varepsilon) \;\wedge\; (\gamma_2, h(\delta)') \vdash_{\overline{\delta}} (\delta_2, \varepsilon) \\
&\iff \quad (\gamma_1, w) \vdash_{\overline{push_A}} (\gamma_1 A, w) \vdash_{\overline{h(\delta)'}} (\delta_1, \varepsilon) \;\wedge \\
&\qquad\quad\; (\gamma_2, push_A h(\delta)') \vdash_{\overline{read_{push_A}}} (\gamma_2, h(\delta)') \vdash_{\overline{\delta}} (\delta_2, \varepsilon) \\
&\iff \quad (\gamma_1, w) \vdash_{\overline{h(\beta)'}} (\delta_1, \varepsilon) \;\wedge\; (\gamma_2, h(\beta)') \vdash_{\overline{\beta}} (\delta_2, \varepsilon)
\end{aligned}$$

Die übrigen Fälle gehen analog. Damit ist die Behauptung $L(\mathcal{M}) = L(\mathcal{M}_1)$ gezeigt. □

Das Arbeiten mit zwei Kellerspeichern entspricht dem menschlichen Arbeiten am Schreibtisch, wo man beim Schreiben einzelne Seiten durch Vor- und Zurückblättern zwischen zwei Stapeln bewegt, und neue Blätter oben auf einen Stapel hinzufügt oder vorhandene dort wegnimmt, wenn man den Text ergänzt oder korrigiert. Diese Vorstellung hat A.Turing[27] formal als ein Maschinenmodell eingeführt.

Eine TURINGMASCHINE hat als Speicher ein in Zellen eingeteiltes lineares endloses Band, auf dem man an jeder Stelle Änderungen, Kürzungen und Erweiterungen vornehmen und auf dem man sich in beiden Richtungen bewegen darf. Nur endlich viele Zellen sind beschriftet. Durch zwei Stapel kann man solche Maschinen simulieren, indem man die links bzw. rechts von der Arbeitsstelle liegende Bandhälfte als Stapel betrachtet. Wir identifizieren daher eine Turingmaschine mit einem Kellerautomaten mit 2 Kellern.

Wie erwartet erkennen rekursive 1-Keller-Automaten, oder einfache 2-Keller-Automaten, mehr als nur kontextfreie Sprachen:

Beispiel 11.12 Sei L die nicht kontextfreie Sprache $\{ww \mid w \in A^*\}$ über $A = \{a, b, c\}$. Als Kelleralphabet nehmen wir $\Gamma := A$, und als Speicher zwei Keller, d.h. $\Gamma^* \times \Gamma^*$ mit *push*- und *pop*-Befehlen für jeden Keller. Wir betrachten die Maschine $(\Gamma, \Gamma, A, M, \varepsilon, \varepsilon)$ mit leeren Anfangskellern und der durch

$$M := \mathit{shift}^* \cdot \mathit{reverse}^* \cdot \mathit{compare}^*$$

definierten Menge von Operationsfolgen, wobei

$$\begin{array}{ll} \mathit{shift} & := \{\, (a, \varepsilon, a^{-1}) \mid a \in A \,\}, \\ \mathit{reverse} & := \{\, (a^{-1}, a, \varepsilon) \mid a \in A \,\}, \\ \mathit{compare} & := \{\, (\varepsilon, a^{-1}, a^{-1}) \mid a \in A \,\}. \end{array}$$

Diese Operationen lassen sich offensichtlich durch Leseoperationen auf der Eingabekomponente und *push*- und *pop*-Operationen auf den Kellern darstellen. Wir wollen mit leerem Speicher akzeptieren, setzen also

$$L(\mathcal{M}) := \{w \in A^* \mid (\varepsilon, \varepsilon, w) \vdash_{\overline{M}} (\varepsilon, \varepsilon, \varepsilon)\}.$$

Man erhält z.B. für $aabcabac \notin L$ die Berechung

$$\begin{array}{lll} (\varepsilon, \varepsilon, aabcabac) & \vdash_{\overline{\mathit{shift}^*}} & (cbaa, \varepsilon, abac) \\ & \vdash_{\overline{\mathit{reverse}^*}} & (\varepsilon, aabc, abac) \\ & \vdash_{\overline{\mathit{compare}^*}} & (\varepsilon, abc, bac). \end{array}$$

Da die anderen Operationsfolgen die Eingabe nicht in der Mitte zerteilen, gibt es keine akzeptierende Berechnung für diese Eingabe.

Wörter in L wie $aabcaabc \in L$ werden dagegen akzeptiert:

$$\begin{array}{lll} (\varepsilon, \varepsilon, aabcabac) & \vdash_{\overline{\mathit{shift}^*}} & (cbaa, \varepsilon, aabc) \\ & \vdash_{\overline{\mathit{reverse}^*}} & (\varepsilon, aabc, aabc) \\ & \vdash_{\overline{\mathit{compare}^*}} & (\varepsilon, \varepsilon, \varepsilon). \end{array}$$

L wird auch durch einen rekursiven 1-Keller-Automaten $\mathcal{M}_r = (\Gamma, A, M_r, \varepsilon)$ erkannt, wobei wir die kontextfreie Menge M_r durch ein rekursives 1-Keller-Programm r definieren, nämlich durch den μ-regulären Ausdruck

$$r := \mathit{shift}^*; \mu x.t$$

über A_Γ, wobei

$$\mathit{shift} := \Sigma_{a \in A}(\mathit{read}_a; \mathit{push}_a) \quad \text{und} \quad t(x) := 1 + \Sigma_{a \in A}(\mathit{pop}_a; x; \mathit{read}_a).$$

Bei der obigen Eingabe $aabcaabc \in L$ operiert $\mathcal{M}_r$ wie folgt:

$$\begin{array}{rll} (\varepsilon, aabcaabc) & \vdash_{\overline{\mathit{shift}}}^{*} & (cbaa, abac) \\ & \vdash_{\overline{\mathit{pop}_c}} & (baa, aabc) \\ & \vdash_{\overline{\mathit{pop}_b}} & (aa, aabc) \\ & \vdash_{\overline{\mu x.t}} & (\varepsilon, bc) \\ & \vdash_{\overline{\mathit{read}_b}} & (\varepsilon, c) \\ & \vdash_{\overline{\mathit{read}_c}} & (\varepsilon, \varepsilon). \end{array}$$

Da $\vdash_{\overline{\mathit{pop}_b; \mu x.t; \mathit{read}_b}} \subseteq \vdash_{\overline{\mu x.t}}$ ist usw., ist die gesamte Rechnung eine Ausführung von r.

Während rekursive 1-Keller-Automaten also mächtiger als normale 1-Keller-Automaten sind, gilt das nicht mehr, wenn die Automaten mindestens zwei Keller haben:

Satz 11.13 Jeder rekursive $n+1$-Keller-Automat $\mathcal{M}^r_{n+1}$ kann durch einen (einfachen) $n+2$-Keller-Automaten $\mathcal{M}_{n+2}$ simuliert werden. Jeder $n+2$-Keller-Automat $\mathcal{M}_{n+2}$ kann durch einen 2-Keller-Automaten $\mathcal{M}_2$ simuliert werden.

Beweis: (Idee) Die erste Behauptung erhält man wie im vorigen Satz 11.11.

Für die zweite Behauptung sei $\mathcal{M}_{n+2}$ ein (nicht rekursiver) Kellerautomat mit $n+2$ Kellerspeichern. Man kann annehmen, daß die $n+2$ Kelleralphabete von $\mathcal{M}_{n+2}$ disjunkt sind. Da jeder Keller Γ_i^* jeweils nur einen endlichen Inhalt $\gamma_i \in \Gamma_i^*$ hat, kann man diese Inhalte im ersten Keller von $\mathcal{M}_2$ "stapeln" und erhält den Inhalt $\gamma = \gamma_1\gamma_2 \cdots \gamma_{n+2}$. Um eine Operation von $\mathcal{M}_{n+2}$ auf Keller Γ_i^* mit $\mathcal{M}_2$ zu simulieren, stapelt man die Inhalte $\gamma_1 \cdots \gamma_{i-1}$ buchstabenweise um auf den zweiten Keller von $\mathcal{M}_2$, wo sie dann gespiegelt als $\delta = \vec{\gamma}_{i-1} \cdots \vec{\gamma}_1$ erscheinen. Auf dem ersten Keller mit Inhalt $\gamma_i \cdots \gamma_n$ führt man mit $\mathcal{M}_2$ nun die Operationen aus, die $\mathcal{M}_{n+2}$ auf Keller Γ_i^* ausführen soll. Anschließend stapelt man mit $\mathcal{M}_2$ wieder den Inhalt des zweiten Kellers auf den ersten zurück, wobei die richtige Reihenfolge automatisch wiederhergestellt wird. Die Simulation des nächsten Befehls von $\mathcal{M}_{n+2}$ durch $\mathcal{M}_2$ kann beginnen.

Da die Kelleralphabete $\Gamma_1, \ldots, \Gamma_{n+2}$ disjunkt sind, erkennt $\mathcal{M}_2$ am obersten Kellersymbol, in welchem Stapel von $\mathcal{M}_{n+2}$ die Simulation gerade arbeitet. $\Box$

Folgerung 11.14 Zu jedem rekursiven $n{+}2$-Keller-Automaten $\mathcal{M}$ gibt es einen einfachen 2-Keller-Automaten $\mathcal{M}'$ mit $L(\mathcal{M}) = L(\mathcal{M}')$.

Damit haben wir einen gewissen Abschluß erreicht: durch rekursive Programme gewinnen wir keine Erhöhung der Rechenleistung gegenüber iterativen Programmen, wenn wir mindestens zwei Stapel als Speicher zur Verfügung haben.

Erkennung rekursiv aufzählbarer Sprachen

In den nächsten beiden Sätzen wird gezeigt, daß jede von einer Turingmaschine erkannte Sprache die Lösungsmenge des Wortproblemes eines Semi-Thue-Systems ist, und daß umgekehrt die Projektion der Lösungsmenge des Wortproblems eines Semi-Thue-Systems die von einer Turingmaschine erkannte Sprache ist.

Satz 11.15 Sei $\mathcal{M} = (\Gamma_1, \Gamma_2, \Sigma, M, S_1, S_2)$ ein 2-Keller-Automat über Σ. Es gibt ein Semi-Thue-System $(\Delta^*, \cdot, E)$ mit $\Sigma \subseteq \Delta$ und ein Wort u über $\Delta \cup \{x\}$ mit

$$L(M) = L(u \Rightarrow_E^* \varepsilon) := \{\, w \in \Sigma^* \mid u[w/x] \Rightarrow_E^* \varepsilon \,\}.$$

Beweis: Wir wollen eine Berechnung mit $\mathcal{M}$, die ein Wort $w \in \Sigma^*$ akzeptiert, durch eine Teilfolge einer Reduktionsfolge

$$u_0 := u[w/x] \Rightarrow_E u_1 \Rightarrow_E \cdots \Rightarrow_E \varepsilon$$

beschreiben, deren Glieder $u_{k_0}, u_{k_1}, u_{k_2} \ldots$ die aufeinander folgenden Konfigurationen in der Rechnung von $\mathcal{M}$ darstellen.

Wir können annehmen, daß die Kelleralphabete Γ_1, Γ_2 und Σ paarweise disjunkt sind. Die reguläre Menge $M \subseteq \Sigma^*_{\Gamma_1 \cup \Gamma_2}$ werde durch den deterministischen endlichen Automaten

$$\mathcal{A} = (Q, \vdash_p, q_0, F)_{p \in \Sigma_{\Gamma_1 \cup \Gamma_2}}$$

erkannt. Zu einer akzeptierenden Berechnung $(S_1, S_2, w) \vdash_\alpha (\varepsilon, \varepsilon, \varepsilon)$ mit $\mathcal{M}$ gibt es eine die Zulässigkeit der Befehlsfolge α kontrollierende Berechnung $q_0 \vdash_\alpha q_e \in F$ mit $\mathcal{A}$.

Sei $\Delta := \Sigma \cup Q \cup \Sigma_{\Gamma_1 \cup \Gamma_2}$. Eine Konfigurantion (γ_1, γ_2, v) von $\mathcal{M}$, die durch eine Rechnung $(S_1, S_2, w) \vdash_\beta (\gamma_1, \gamma_2, v)$ aus der Anfangskonfiguration erreicht wird, beschreiben wir durch das Wort $q_i \gamma_1 \gamma_2 v q_0 \in \Delta^+$, wobei q_i durch $q_0 \vdash_\beta q_i$ eindeutig bestimmt ist.

Sei $u(x) := q_0 S_1 S_2 x q_0$. Für E wählen wir folgende definierenden Relationen:

$\mathcal{A}$	E			
$q_i \vdash_{read_a} q_j$	q_i	$\to$	$q_j\, read_a$	
	$read_a\, A$	$\to$	$A\, read_a,$	falls $A \in \Gamma_1 \cup \Gamma_2,$
	$read_a\, a$	$\to$	ε	
$q_i \vdash_{push_A} q_j$	q_i	$\to$	$q_j\, push_A$	
	$push_A\, B$	$\to$	$B\, push_A,$	falls $A \in \Gamma_2,\ B \in \Gamma_1,$
	$push_A\, B$	$\to$	$A\, B,$	falls $A, B \in \Gamma_k$ oder $B \in \Gamma_2 \cup \Sigma \cup \{q_0\},$
$q_i \vdash_{pop_A} q_j$	q_i	$\to$	$q_j\, pop_A,$	
	$pop_A\, B$	$\to$	$B\, pop_A,$	falls $A \in \Gamma_2$ und $B \in \Gamma_1,$
	$pop_A\, B$	$\to$	$\varepsilon,$	falls $A = B \in \Gamma_1 \cup \Gamma_2,$
$q_i \in F$	$q_i\, q_0$	$\to$	$\varepsilon.$	

Mit diesen Regeln können wir am linken Rand eines Worts $u_{k_i} = q_i \gamma_1 \gamma_2 v q_0$ eine durch $\mathcal{A}$ erlaubte Instruktion aus $\Sigma_{\Gamma_1 \cup \Gamma_2}$ erzeugen und dann nach rechts 'durch u_{k_i} schieben', um an γ_i oder v die Ausführung der Instruktion durch $\mathcal{M}$ zu simulieren, was zu $u_{k_{i+1}}$ führt. Die Instruktionen werden in der gleichen Reihenfolge ausgeführt, in der sie erzeugt wurden. Man überlege sich, daß daher gilt:

$$\exists \beta \in \Sigma^*_{\Gamma_1 \cup \Gamma_2} [q_0 \vdash_\beta q_i \wedge (S_1, S_2, w) \vdash_\beta (\gamma_1, \gamma_2, v)] \quad \Longleftrightarrow \quad u[w/x] \Rightarrow^*_E q_i \gamma_1 \gamma_2 v q_0$$

Wegen der letzten Regel von E ist folglich $w \in L(\mathcal{M})$ genau dann, wenn $u[w/x] \Rightarrow^*_E \varepsilon$. $\square$

Satz 11.16 Sei $(\Sigma^*, \cdot, E)$ ein Semi-Thue-System und $u(\vec{x}), v(\vec{x}) \in (\Sigma \cup \{x_1, \ldots, x_m\})^*$. Es gibt eine Turingmaschine $\mathcal{M}$ mit

$$L(u \Rightarrow^*_E v)_1 = \{\, w_1 \in \Sigma^* \mid \exists w_2, \ldots, w_m \in \Sigma^*\ u(\vec{w}) \Rightarrow^*_E v(\vec{w}) \,\} = L(\mathcal{M}).$$

Beweis: (Skizze) Nach dem früher Bewiesenen genügt es zu zeigen, daß die Sprache $L(u \Rightarrow^*_E v)_1$ von einem $2m+2$-Keller-Automaten $\mathcal{M}$ erkannt wird. Die Wörter u, v enthalten nur endlich viel Information, die (u.a.) in den Zuständen der 'endlichen Kontrolle' $\mathcal{A}$ von $\mathcal{M}$ gespeichert wird. Sei $w_1 \in \Sigma^*$ das Eingabewort. $\mathcal{M}$ schreibt w_1 in den ersten Keller und schreibt dann willkürlich (mit nicht-deterministischen Übergängen) Wörter $w_2, \ldots, w_m$ in die Keller $3, 5, \ldots, 2m-1$. Anschließend werden zwei benachbarte Keller $2k-1$ und $2k$ für $k \leq m$ dazu benutzt, im Wort w_k zu lesen (vgl. die Simulation einer $2n+2$-Keller-Maschine durch eine 2-Keller-Maschine).

Auf den vorletzten Keller schreibt $\mathcal{M}$ nun, unter Verwendung der gespeicherten Information über $u(\vec{x})$ und durch Kopieren der $w_1, \ldots, w_m$ das Wort $u(\vec{w})$. Da E nur endlich viel

Information enthält, kann auch E in den Zuständen von $\mathcal{A}$ gespeichert werden. $\mathcal{M}$ wählt ein $(v, v') \in E$ und stellt durch Lesen von $u(\vec{w})$ (unter Benutzung der Keller $2m+1$ und $2m+2$) fest, ob v ein Teilwort von $u(\vec{w})$ ist. Falls ja, ersetzt $\mathcal{M}$ ein Vorkommen von v durch v' (unter Benutzung der Keller $2m+1$ und $2m+2$), wobei das Ergebnis anschließend im Keller $2m+1$ steht und Keller $2m+2$ wieder leer ist. Damit ist eine Anwendung $u(\vec{w}) \Rightarrow_E u_1$ simuliert. Mit der gespeicherten Information über $v(\vec{x})$ und den $\vec{w}$ in den Anfangskellern prüft $\mathcal{M}$ nun, ob $u_1 = v(\vec{w})$ Falls ja, stoppt $\mathcal{M}$ und akzeptiert w_1. Falls nein, wird versucht, einen Schritt $u_1 \Rightarrow_E u_2$ in entsprechender Weise zu simulieren. Falls keine Regel von E anwendbar ist, stoppt $\mathcal{M}$, ohne w_1 zu akzeptieren. □

Eine Sprache L heißt REKURSIV AUFZÄHLBAR, wenn es eine Turingmaschine $\mathcal{M}$ gibt, die L akzeptiert.

Der Name kommt daher, daß dies auch genau die Sprachen sind, die durch Turingmaschinen AUFGEZÄHLT werden können: enthält die Anfangskonfiguration eine Zahl n in Keller 1, so soll die Maschine daraus ein Wort $w_n \in \Sigma^*$ bilden, d.h. mit w_n auf Keller 1 anhalten. Dann wird $L = \{ w_n \mid n \in \mathbb{N} \}$ durch die Maschine aufgezählt.

Automatenhierarchie

Von den vier Sprachklassen der Chomsky-Hierarchie haben wir damit für drei eine Charakterisierung durch erkennende Automaten. Für die noch fehlenden Klasse der kontextsensitiven Sprachen braucht man einen neuen Begriff: Der PLATZBEDARF einer 2-Kellermaschine $\mathcal{M}$ bezüglich eines Eingabeworts $w \in \Sigma^*$ ist die maximale Anzahl der Symbole auf den Kellern, die während einer akzeptierenden Berechnung mit Eingabe w erreicht wird:

$$Platz(w, \mathcal{M}) := \max\{ |\alpha| + |\beta| \mid (S_1, S_2, w) \vdash^*_{\mathcal{M}} (\alpha, \beta, v) \vdash^*_{\mathcal{M}} (\varepsilon, \varepsilon, \varepsilon) \}$$

Eine Turingmaschine $\mathcal{M}$ heißt LINEAR BESCHRÄNKT, wenn es eine Zahl $c \in \mathbb{N}$ gibt, so daß $Platz(w, \mathcal{M}) \leq c \cdot |w|$ für alle $w \in L(\mathcal{M})$ bis auf endlich viele Ausnahmen.

Satz 11.17 Sei L eine formale Sprache über Σ. Dann gilt:

(i) $L \in \mathcal{L}_0$ genau dann, wenn L von einer Turingmaschine erkannt wird.

(ii) $L \in \mathcal{L}_1$ genau dann, wenn L von einer linear beschränkten Turingmaschine erkannt wird.

(iii) $L \in \mathcal{L}_2$ genau dann, wenn L durch einen nicht-deterministischen Kellerautomaten erkannt wird.

(iv) $L \in \mathcal{L}_3$ genau dann, wenn L von einem endlichen Automaten erkannt wird.

Beweis: (i) Das wurde mit Satz 11.15 und Satz 11.16 gezeigt.

(ii) $\Rightarrow$: Da $\Longrightarrow_G$ ein Semi-Thue-System ist, genügt es, den Beweis von Satz 11.16 geeignet zu modifizieren. Wir betrachten nur den Fall, wo $\varepsilon \notin L(G)$ ist. Dann ist in einer Ableitung $S \Longrightarrow_G \alpha_1 \Longrightarrow_G \ldots \Longrightarrow_G \alpha_n \Longrightarrow_G w$ jeweils $|\alpha_i| \leq |w|$. Zunächst beschreiben wir einen 4-Keller-Automaten mit beschränktem Platzbedarf, der $L(G)$ akzeptiert.

Mit Keller 1 und 2 wird eine G-Ableitung simuliert. Zu Beginn der Rechnung schreibt man S in Keller 1. Enthält Keller 1 die Satzform α_i und ist Keller 2 leer, so stapelt man α_i nach Keller 2 um und sucht dabei nach einem Teilwort α, das die linke Regelseite einer Regel $\alpha \to \beta \in P$ ist. Beim Umstapeln von α_i nach Keller 2 darf nun das Teilwort α durch β ersetzt werden. Anschließend stapelt man den Inhalt von Keller 2 zurück nach Keller 1, so daß dort α_{i+1} steht. Da der Kellerautomat nicht-deterministisch arbeitet, kann so jede Regelanwendung versucht werden.

Diese Simulation von G-Ableitungen auf den Kellern 1 und 2 wird mit den Kellern 3 und 2 so kontrolliert, daß Keller 1 und 2 zusammen höchsens $|w|$ viele Symbole enthalten. Dazu wird zu Beginn der Rechnung die Eingabe w in Keller 3 kopiert. Während des Umstapelns von Keller 1 nach Keller 2 wird der Inhalt von Keller 3 nach Keller 4 umgeschichtet; wenn Keller 3 leer ist, also $|w|$ Symbole in Keller 2 geschrieben wurden, muß auch Keller 1 geleert sein; ist das nicht der Fall, bricht die Rechnung ab, ohne die Eingabe zu akzeptieren. Wird Keller 2 nach Keller 1 umgeschichtet, so macht man das auch mit Keller 4 und Keller 3. Da G keine verkürzenden Regeln hat, kann jede G-Ableitung von w so simuliert werden. Der Gesamtinhalt der 4 Keller besteht aus höchstens $2 \cdot |w|$ vielen Symbolen.

Nach 11.13 kann dieser 4-Keller-Automat durch eine Turingmaschine $\mathcal{M}$ simuliert werden. Da diese Simulation keinen zusätzlichen Platz erfordert, ist $\mathcal{M}$ linear beschränkt.

$\Leftarrow$: (Idee) Zur Vereinfachung sei wieder $\varepsilon \notin L(\mathcal{M})$ für die gegebene Turingmaschine $\mathcal{M}$. Man kann annehmen, daß sich $\mathcal{M}$ zuerst die Eingabe w in den Keller 1 kopiert und dann den maximal erlaubten Platz $c \cdot |w|$ durch weitere Symbole P markiert, so daß $L(M) = \{\, w \in \Sigma^+ \mid (wP^{(c-1)|w|}, \varepsilon) \vdash_M (\varepsilon, \varepsilon) \,\}$ nur noch die beiden Keller manipuliert, ohne die Summe der Symbole in den Kellern je wieder zu erhöhen. Die Konfigurationen von $\mathcal{M}$ bestehen dann aus den Inhalten γ_1 und γ_2 der beiden Keller und dem Zustand q, in dem sich der endliche Automat befindet, der die Zulässigkeit der ausgeführten Befehlsfolge kontrolliert. Eine akzeptierende $\mathcal{M}$-Berechnung

$$(wP^{(c-1)|w|}, \varepsilon) \;\equiv\; (\gamma_{1,1}, \gamma_{2,1}) \;\vdash\; (\gamma_{1,2}, \gamma_{2,2}) \;\vdash\; \ldots \;\vdash\; (\varepsilon, \varepsilon)$$

wird in umgekehrter Reihenfolge als eine Folge von "Konfigurationswörtern"

$$q_n \;\Longrightarrow_G\; \cdots \;\Longrightarrow_G\; \overleftarrow{\gamma_{2,1}} q_1 \gamma_{2,2} \;\Longrightarrow_G\; \overleftarrow{\gamma_{1,1}} q_1 \gamma_{2,1} \;\equiv\; P^{(c-1)|w|} \overleftarrow{w} q_0$$

beschrieben, wobei $q_0, q_1, \ldots$ die durchlaufenen Zustände der endlichen Kontrolle sind und $\overleftarrow{\alpha}$ das Spiegelbild von α ist. Dies kann man als Ableitung einer kontextsensitiven Grammatik G auffassen, da die Summe der Symbole in den Kellern (nach der Eingabephase) nie erhöht wird. Die Übergänge von $\mathcal{M}$ werden zu lokalen, nicht verkürzenden Veränderungen von Konfigurationswörtern:

Konfigurationsänderung	Grammatikregel
$p,(A\gamma_1,\gamma_2) \vdash_{pop_A;\,push_{A'}} q,(A'\gamma_1,\gamma_2)$	$A'q \to Ap$
$p,(A\gamma_1,\gamma_2) \vdash_{pop_A;\,push_B} q,(\gamma_1,B\gamma_2)$	$qB \to Ap$
$p,(\gamma_1,B\gamma_2) \vdash_{pop_B;\,push_{B'}} q,(\gamma_1,B'\gamma_2)$	$qB' \to pB$
$p,(\gamma_1,B\gamma_2) \vdash_{pop_B;\,push_A} q,(A\gamma_1,\gamma_2)$	$Aq \to pB$
$p,(A\gamma_1,\varepsilon) \vdash_{pop_A} q,(\gamma_1,\varepsilon)$	$q \to Ap$
$p,(\varepsilon,B\gamma_2) \vdash_{pop_B} q,(\varepsilon,\gamma_2)$	$q \to pB$

Nimmt man noch $S \to q_n$ für akzeptierende Zustände q_n und entsprechende Regeln für $push_X; pop_Y$-Schritte hinzu, so hat man eine Grammatik $G = (N, \Sigma, P, S)$, die gerade die Umkehrungen der akzeptierenden Berechnungen von $\mathcal{M}$ erzeugt. Also ist $L(G) = L(\mathcal{M})$, und nach 11.8 ist G eine kontextsensitive Grammatik.

(iii) In Satz 7.8 haben wir kontextfreie Sprachen durch Maschinen charakterisiert.

(iv) Das besagen der Darstellungssatz 4.5 von Kleene und Proposition 4.8. □

Die in Satz 11.17 festgestellte Entsprechung zwischen Klassen von Phrasenstrukturgrammatiken und erkennenden Automaten kann man auf eine Reihe weiterer Grammatikarten ausdehnen, wie es für die linearen Grammatiken schon in Abschnitt 5.4 getan wurde.

11.4 Abgeschlossenheitseigenschaften

Man kann zeigen, daß zu jeder Turingmaschine eine äquivalente DETERMINISTISCHE TURINGMASCHINE existiert, d.h. eine, bei der in jeder Konfiguration (γ_1, γ_2, v) die Nachfolgekonfiguration eindeutig durch die obersten Kellersymbole, das erste Symbol der Eingabe und den Zustand der endlichen Kontrolle bestimmt ist. Eine Sprache $L \subseteq \Sigma^*$ heißt ENTSCHEIDBAR oder REKURSIV, wenn sie von einer deterministischen Turingmaschine $\mathcal{M}$ akzeptiert wird, die für jedes $w \in \Sigma^*$ nach endlich vielen Schritten stoppt, d.h. in eine Konfiguration gerät, die keine Nachfolgekonfiguration hat.

Beachte, daß eine Turingmaschine, die L erkennt, für Eingaben $w \notin L$ im allgemeinen eine endlose Rechnung ausführt, so daß zu keinem Zeitpunkt klar ist, ob $w \in L$ oder $w \notin L$.

Sei $\mathcal{L}_{rek}$ die Menge der entscheidbaren Sprachen über Σ. Ohne Beweis geben wir noch einige Abschlußeigenschaften der kontextsensitiven, rekursiven und rekursiv aufzählbaren Sprachklassen an:

Satz 11.18 Sei $L \subseteq \Sigma^*$ mit $|\Sigma| \geq 2$.

(i) $\mathcal{L}_1 \subset \mathcal{L}_{rek} \subset \mathcal{L}_0$.

(ii) (Satz von E.Post) $L \in \mathcal{L}_{rek}$ genau dann, wenn $L \in \mathcal{L}_0$ und $\Sigma^* - L \in \mathcal{L}_0$.

(iii) $\mathcal{L}_1, \mathcal{L}_{rek}$ und $\mathcal{L}_0$ sind abgeschlossen unter den regulären Operationen $\cdot$ und * und unter den mengentheoretischen Operationen $\cup$ und $\cap$.

(iv) $\mathcal{L}_1$ und $\mathcal{L}_{rek}$ sind unter Komplementbildung abgeschlossen, $\mathcal{L}_0$ aber nicht.

11.5 Unentscheidbarkeitseigenschaften

Wir haben im Beweis von Satz 11.11 gesehen, wie man mit einem 2-Keller-Automaten das Programm eines (rekursiven) 1-Kellerautomaten INTERPRETIERT. Man kann solche Kellerautomaten angeben, die *jedes* Programm einer Turingmaschine interpretieren können, sogenannte UNIVERSELLE TURINGMASCHINEN.

Verwendet das Programme M_1 einer Maschine als Eingabe anderer Maschinen M_2, so kann man insbesondere die Maschine M_2 dazu benutzen, Fragen über die Maschine M_1 zu beantworten (als positiv lösbar 'akzeptieren' oder ablehnen).

Turingmaschinen M sind, als Programme von 2-Kellermaschinen betrachtet, Wörter über einem geeigneten Alphabet Σ. Das HALTEPROBLEM FÜR TURINGMASCHINEN ist das Problem, von beliebigen $M \in \Sigma^*$ festzustellen, ob M bei mindestens einer Eingabe zu einer Endkonfiguration übergehen kann. Man kann nun zeigen:

Satz 11.19 Das Halteproblem für Turingmaschinen ist rekursiv unlösbar: es gibt keine Turingmaschine T, die für jede Turingmaschine M feststellen kann, ob M bei mindestens einer Eingabe in eine Endkonfiguration übergehen kann.

Ebenso sind die Fragen, ob M bei einer vorgegebenen Eingabe oder bei allen Eingaben in eine Endkonfiguration übergehen kann, unentscheidbar. Wir wollen das hier nicht beweisen, sondern nur eine Konsequenz über kontextfreie Grammatiken ableiten, daß man nämlich schon sehr einfache Fragen über $L(G)$ nicht an der Grammatik G ablesen kann. Sogar für die Teilklasse der linearen Grammatiken gilt schon:

Satz 11.20 Es gibt kein Verfahren, das für jede lineare Grammatik G über Σ feststellen kann, ob $L(G) = \Sigma^*$ ist.

Beweis: Man konstruiert aus einer Turingmaschine M eine lineare Grammatik G_M mit

$$L(G_M) = \Sigma^* \iff L(M) = \emptyset. \tag{67}$$

Ein Entscheidungsverfahren für das Problem $L(G) = \Sigma^*$ würde also ein Entscheidungsverfahren für das Halteproblem von Turingmaschinen liefern, was unmöglich ist.

Sei $M = (\Gamma_1, \Gamma_2, \Sigma, M, S_1, S_2)$ ein 2-Keller-Automat mit Eingabealphabet Σ. Wir können annehmen, daß $\mathcal{M}$ deterministisch ist und zusätzlich die Eingabe $w \in \Sigma^*$ zuerst in Keller 2 kopiert, also jede Rechnung die Form

$$(S_1, S_2, w) \vdash^* (S_1, w, \varepsilon) \vdash^* (\varepsilon, \varepsilon, \varepsilon)$$

hat, so daß man das Eingabeband ignorieren kann. Sei $\mathcal{A} = (Q, \vdash_p, q_0, F)_{p \in \Sigma_\Gamma}$ mit $\Gamma = \Gamma_1 \cup \Gamma_2$ ein deterministischer endlicher Automat mit $M = L(\mathcal{A}) \subseteq \Sigma_\Gamma^*$, die endliche Kontrolle von $\mathcal{M}$ (vgl. den Beweis von 11.15).

Eine Konfigurationsbeschreibung von $\mathcal{M}$ gibt die Kellerinhalte und den Zustand der endlichen Kontrolle $\mathcal{A}$ von $\mathcal{M}$ (im zweiten Teil der Berechnung) wieder. Wir schreiben $ID(M) := \Gamma_1^* Q \Gamma_2^*$ für die Gesamtheit der Konfigurationsbeschreibungen.

Ein Wort v über $\Delta := \Gamma \cup Q \cup \{\#\}$ ist eine gültige Berechnungsfolge von M, wenn

(i) $v = w\overleftarrow{w}$ für ein $w = w_1 \# w_2 \# \dots \# w_n \# \in N := (ID(M) \cdot \{\#\})^+$,

(ii) w_1 ist eine Anfangskonfiguration von M, d.h. $w_1 \in q_0 \Sigma^*$,

(iii) w_n ist eine Endkonfiguration von M, d.h. $w_n \in \Gamma^* F \Gamma^*$,

(iv) $w_1, \dots, w_n \in ID(M)$ ist die Folge von Konfigurationsbeschreibungen der in einer $\mathcal{M}$-Berechnung $(S_1, u, \varepsilon) \vdash_\beta (\varepsilon, \varepsilon, \varepsilon)$, mit $\beta \in M$ und $u \in \Sigma^*$, durchlaufenen Konfigurationen.

Lemma 11.21 Die Menge $invalid(M)$ ist eine lineare Sprache.

Beweis: (Skizze) Ungültige Berechnungsfolgen sind Zeichenreihen über Δ, die eine der obigen vier Bedingungen an gültige Berechnungsfolgen verletzen, d.h. in

$$\Delta^* - (N\overleftarrow{N} \ \cap \ q_0\Sigma^*\#\Delta^* \ \cap \ \Delta^*\#\Gamma^*F\Gamma^* \ \cap \ O)$$

liegen, wobei

$$O := \{\, w_1\#w_2\#\dots\#w_n\#\#\overleftarrow{w_n}\#\dots\#\overleftarrow{w_1} \mid w_i \in ID(M), n \in I\!N,\ w_1 \vdash_M \cdots \vdash_M w_n \,\}.$$

Klar ist, daß mit N auch $N\overleftarrow{N}$, $q_0\Sigma^*\#\Delta^*$ und $\Delta^*\#(\Gamma^*F\Gamma^*)$ regulär sind. Damit ist die Vereinigung ihrer Komplemente regulär, also linear. Es bleibt zu zeigen, daß auch das Komplement von O linear ist.

Das Komplement von O ist eine lineare Sprache, wenn es von einer 1-Kellermaschine $\mathcal{K}$ akzeptiert wird, die nur eine Umkehrung vom Schreiben in den Stapel zum Lesen aus dem Stapel macht. (Übung!) Ein solches $\mathcal{K}$ gibt es für $\Delta^* - O$: Während des Lesens von $w_i \in ID(M)$ schreibt $\mathcal{K}$ *die* Nachfolgekonfiguration $M(w_i)$ als $\overleftarrow{M(w_i)}\#$ in ihren Kellerspeicher (das geht, da w_i und $M(w_i)$ sich nur lokal, in einem Teilwort der Länge ≤ 3, unterscheiden). Sobald zwei aufeinanderfolgende Trennsymbole $\#$ in der Eingabe auftreten, entfernt $\mathcal{K}$ die oberste Nachfolgerkonfiguration vom Stapel. Dann wird durch Lesen der Resteingabe $\overleftarrow{w_n}\# \cdots \#\overleftarrow{w_2}$ und Entfernen des Kellerinhalts festgestellt, ob $\overleftarrow{w_i} = \overleftarrow{M(w_{i-1})}$ für $i = n, \ldots, 2$ der Fall ist. Wenn ja, ist die gelesene Eingabe v eine gültige Berechnungsfolge von $\mathcal{M}$ und wird abgelehnt; sonst wird v akzeptiert. □

Sei G_M die lineare Grammatik mit $L(G_M) = invalid(M)$. Dann ist $L(M) = \emptyset$ genau dann, wenn $\mathcal{M}$ kein Wort akzeptiert, wenn es also keine gültige Berechnung von $\mathcal{M}$ gibt, und das heißt, wenn $\Delta^* = invalid(M) = L(G_M)$ ist. □

Für die Klasse $\mathcal{L}_0$ der rekursiv aufzählbaren Sprachen L kann man *überhaupt keine* nicht triviale Eigenschaft der Sprache $L = L(G)$ an der Typ-0-Grammatik G ablesen:

Satz 11.22 (Satz von Rice) Sei $E \subseteq \mathcal{L}_0$ eine nicht-triviale Eigenschaft rekursiv aufzählbarer Sprachen, d.h. $\emptyset \neq E \neq \mathcal{L}_0$. Es gibt kein Verfahren, mit dem man für jede Phrasenstrukturgrammatik G feststellen könnte, ob $L(G) \in E$.

Einen Beweis findet man z.B. in Hopcroft/Ullman[11]. Die Aussage des Satzes gilt entsprechend für Turingmaschinen: am Programm M einer Turingmaschine kann man nicht erkennen, ob die von M akzeptierte Sprache die Eigenschaft E hat, also ob $L(M) \in E$ ist. Da man in jeder heute üblichen Programmiersprache Turingmaschinen simulieren kann, darf man sich also keinen Illusionen darüber hingeben, was man automatisch über die Bedeutung beliebiger Programme am Programmtext ablesen könnte.

Aufgaben

Aufgabe 11.1 Analog zur Definition der rekursiven endlichen Automaten in 7.1 definiere man einen rekursiven Kellerautomaten als eine endliche Menge von Kellerautomaten, die sich gegenseitig aufrufen dürfen.

Zeige, daß diese Systeme von Kellerautomaten dieselben Sprachen erkennen wie die rekursiven Kellerautomaten nach Definition 11.10.

Aufgabe 11.2 Zeige oder widerlege die Umkehrung von Satz 11.11:

Jede Sprache, die von einem einfachen 2-Kellerautomaten erkannt wird, wird auch von einem rekursiven 1-Keller-Automaten erkannt.

Bem.: Nach Satz 7.8 wird jede Sprache, die von einem einfachen 1-Keller-Automat erkannt wird, auch von einem rekursiven endlichen Automaten erkannt wird.

12 Nachwort

Automaten und Programmierung

Damit nicht der Eindruck entsteht, die hier betrachteten Automaten seien sinnlos weit von gebräuchlichen Rechenmaschinen entfernt, haben wir den Zusammenhang zur modularen Programmierung stärker als sonst üblich betont. Dazu haben wir die Programmkonstrukte Seqenzierung ; oder $\cdot$, nicht-deterministische Alternative $+$, Iteration * und Rekursion μ behandelt, aber nicht etwa die Parallelausführung $||$. Iterative Programme über einer Menge Σ von Grundbefehlen haben wir durch reguläre Ausdrücke über Σ formuliert und rekursive Programme durch μ-reguläre Ausdrücke, wozu die regulären Ausdrücke um einen kleinsten-Fixpunkt-Operator μ erweitert wurden. Die Programme wurden als Erkennungsverfahren behandelt, die keine Ausgabewerte berechnen, wenn man von dem Boole'schen Wert, ob das Eingabewort akzeptiert wird oder nicht, absieht.

Wir haben die Mächtigkeit von Programmen ohne und mit Rekursion über einem endlichen Speicher und über einem bzw. mehreren Stapelspeichern untersucht. Bei einem endlichen Arbeitsspeicher werden die Grundbefehle $a \in \Sigma$ durch Übergangsrelationen $\vdash_a$ auf der endlichen Menge Q der möglichen Speicherinhalte interpretiert; bei einem Stapelspeicher werden spezielle Grundoperationen $push_A, pop_A$ als Vergrößern und Verkleinern des Stapels um A interpretiert, d.h. als Übergangsrelationen $\underset{push_A}{\vdash}$, $\underset{pop_A}{\vdash}$ auf der unendlichen Menge Q^* der möglichen Stapelinhalte.

Auf der Seite der Maschinen entsprach die Iteration * der Rückkopplung, die Rekursion μx dem Selbstaufruf. Die Interpretation rekursiver Programme erfordert einen Stapel zur Verwaltung der Rücksprungadressen. Daraus ergab sich eine Einteilung der Mächtigkeit von Programmen zur Erkennung von Wortmengen in drei Klassen:

Programme	Maschine	Wortmenge	Grammatik
iterativ, endlicher Speicher	endlicher Automat	regulär	rechtslineares Gleichungssystem
rekursiv, endlicher Speicher	rekursiver endlicher Automat	μ-regulär,	reguläres Gleichungssystem,
iterativ, 1 Stapelspeicher	1-Keller-Automat	kontextfrei	kontextfrei
rekursiv, ≥ 1 Stapelspeicher	rekursiver 1-Keller-Automat	rekursiv aufzählbar,	
iterativ, ≥ 2 Stapelspeicher	2-Keller-Automat, Turingmaschine	STS-Lösungsmenge	allg. Phrasenstrukturgrammatik

Neben diesen grundsätzlichen Unterscheidungen zwischen iterativen und rekursiven Programmen kann man feinere Unterschiede beachten, insbesondere die Form der Rekursion. Wir haben gesehen, daß Rechts- oder Endrekursion $\mu x((a;x)+b)$ durch Iteration simuliert werden kann und daß lineare Rekursion $\mu x((a;x;)b+c)$ ein umfassenderer, aber echter Spezialfall der allgemeinen Rekursion $\mu x.r$ ist.

Den Spezialfall der deterministischen Alternative, haben wir nur bei endlichen Automaten betrachtet. Es sei bemerkt, daß man auch für Kellerautomaten zwischen deterministischen und nicht deterministischen Automaten unterscheidet. Im Gegensatz zur Situation bei endlichen Automaten lassen sich aber nicht alle Sprachen, die von nicht-deterministischen Kellerautomaten erkannt werden, auch von deterministischen Kellerautomaten erkennen.

Grammatiken und logische Sprachdefinitionen

Wir haben kontextfreie Grammatiken als induktive Definitionen von Ausdrucksmengen betrachtet, wie sie in der Mathematik und Informatik üblich sind. Wir erwähnen einige Züge dieser Grammatiken, die z.B. für die Beschreibung natürlicher Sprachen stören.

(i) Bei diesen 'positiven' induktiven Definitionen (z.B. der Formeln der Logik oder der Programme einer Programmiersprache) wurden nur *gewisse* monotone Operatoren zugelassen: $\cup$, $\cdot$, *. Es gibt keinen triftigen Grund, andere monotone Mengenoperationen wie $\cap$ auszuschließen, nur weil sie keine natürliche Deutung als Programmkonstrukte haben.

(ii) Der technische Begriff der 'kontextfreien' Grammatikregel verdeckt einen meines Erachtens wichtigeren Begriff: die syntaktischen Kategorien $A \subseteq \Sigma^*$ einer Grammatik G über Σ sind KONTEXTUNABHÄNGIG, wenn es für jedes $w \in \Sigma^*$ nur von grammatischen Eigenschaften der echten Teilwörter von w abhängt, ob $w \in A$ ist.

Da mit E auch $\Sigma^* - E$ kontextunabhängig ist, ist nicht jede kontextfreie Menge auch kontextunabhängig. Aber alle kontextunabhängigen Sprachen erlauben ein Erkennungsverfahren durch Rekursion über die Teilwortrelation, also die im Algorithmus von Cocke, Younger und Kasami verwendete Methode des dynamischen Programmierens, wobei Zwischenergebnissen in einer Tabelle abgespeichert werden.

(iii) Es ist ein gemeinsamer Zug von Programmiersprachen, logischen Sprachen und natürlichen Sprachen, daß die Gesamtheit der Ausdrücke *nicht* kontextfrei definierbar ist, wenn man keine leeren Aufbauschritte — wie die Quantifizierung nicht auftretender Variablen — erlaubt.

Das genauere Bild ist vielmehr: die Klasse L der syntaktisch korrekten Ausdrücke ist eine Aussonderung aus einer kontextfreien Sprache:

$$L = \{w \in L(G) \mid \varphi(A(w))\} = L(G) \cap \text{Test}$$

Hierbei ist $A(w)$ eine Analyse von w, d.h. eine syntaktische Struktur gemäß G, und φ ist eine Eigenschaft von syntaktischen Strukturen.

Bei Programmiersprachen wird zum Beispiel getestet, ob der Analysebaum mit Typen dekoriert werden kann oder ob Variablen vor ihrem Gebrauch deklariert wurden; bei logischen Sprachen sollte getestet werden, ob quantifizierte Variablen im Wirkungsbereich des Quantors tatsächlich vorkommen; bei natürlichen Sprachen müssen anaphorische Bezüge auflösbar sein usw.

Eine Grammatik sollte es erlauben, solche NEBENBEDINGUNGEN AN DIE ANWENDBARKEIT VON AUFBAUSCHRITTEN zu formulieren. Zusätzlich könnte man von Elementen *abhängige* syntaktische Kategorien brauchen:

$$a \in A \in Cat \quad \Longrightarrow \quad B(a, A) \in Cat.$$

Bei formalen Sprachen wäre etwa

$$x \in frei(\varphi) \in Formel(x, y) \Longrightarrow \forall x\varphi \in Formel(y)$$

eine solche Regel mit Nebenbedingung. Bei natürlichen Sprachen wäre eine Nebenbedingung etwa, daß Nominalphrasen nur in *ungesättigte* Verbalkomplexe eingesetzt werden dürfen:

$$\forall(\lambda x : A.\varphi(x)) \quad \text{ist erlaubt, aber} \quad \forall(\lambda x : A.\varphi(c)) \quad \text{nicht.}$$

Z.B. ((jeder Affe) (sitzt auf dem Baum)) im Unterschied zu ((jeder Affe) (Emil sitzt auf dem Baum)).

Solange die Nebenbedingung, die die Ausdrücke erfüllen müssen, nur *beschränkt* viel Information über die Ausdrücke erfordern, kann man diese Information in Merkmalen an den Kategorien speichern.

(iv) In den von Chomsky eingeführten Sprachdefinitionen durch Phrasenstrukturgrammatiken sind alle syntaktischen Kategorien *einstellige Prädikate*, also Mengen oder Eigenschaften von Zeichenreihen sind. Dieser Ansatz berücksichtigt *Relationen* zwischen Teilausdrücken nicht.

Einige Relationen, die beim Sprachaufbau eine Rolle spielen, kann man durch *Merkmale* wie Kasus, Numerus, Tempus in natürlichen Sprachen oder Typen in formalen Sprachen darstellen: im wesentlichen wird dabei eine Kongruenzrelation (z.B. daß das finite Verb und die Subjektnominalphrase eines Satzes im Numerus und Person übereinstimmen) auf die Gleichheit von Merkmalwerten der Teilausdrücke reduziert.

Es ist aber nicht offensichtlich, daß dies stets möglich oder günstig im Sinne der Ökonomie der Sprachbeschreibung oder -verarbeitung ist. Die Grammatik einer natürliche Sprache sollte sicherlich über verschiedene Beziehungen von Teilsätzen Rechenschaft geben, oder über die Zusammengehörigkeit *verschobener* Teilausdrücke.

Eine logische Sprachdefinition könnte solche komplexen Relationen induktiv aus mehrstelligen Grundrelationen, z.B. der Beziehung zwischen den Formvarianten der Wörter, definieren. Neben solchen mehrstelligen Grundrelationen könnte es sinnvoll sein, in den definierenden Bedingungen der syntaktischen Kategorien auch die (beschränkte) Quantifizierung über Individuenvariablen (für Zeichenreihen) sowie Konjunktion und –in eingeschränkter Form– Negation zu verwenden. Die Existenz von kleinsten Fixpunkten der entsprechenden induktiven Definition wird dadurch nicht ausgeschlossen.

Ein Grammatiktyp, der die Formulierung von Nebenbedingungen erlaubt, sind Definite Clause Grammars, die sozusagen kontextfreie Grammatiken sind, bei denen die Nonterminale wie Prädikate über die in erststufigen Termen gespeicherte Information behandelt werden. Untersuchungen, die eine Hierarchie der definierbaren Sprachen entsprechend der logischen Form der Nebenbedingungen angeben, scheint es nicht zu geben.

Mit den Methoden der Theorie endlicher Strukturen sollte es möglich sein, die Komplexität von Erkennungsalgorithmen an der Form der logischen Sprachdefinition abzulesen.

Literatur

[1] A. V. Aho and J. D. Ullman. *The Theory of Parsing, Translation and Compiling, Vol. I: Parsing.* Prentice Hall, Englewood Cliffs, N.J., 1972.

[2] M. Böttner, C. Brink, and K. Britz. Peirce algebras. Manuscript. Research School of Social Sciences. The Australian National University, Canberra, 1989.

[3] W. Brauer. *Automatentheorie.* B.G.Teubner, 1984.

[4] C. Brink. Boolean modules. *Journal of Algebra*, 71:291–313, 1981.

[5] J. H. Conway. *Regular Algebra and Finite Machines.* Chapman and Hall, London, 1971.

[6] J. Earley. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102, 1970.

[7] J. Goldstine. Formal languages and their relation to automata. What Hopcroft & Ullman didn't tell us. In R. v. Book, editor, *Formal Language Theory. Perspectives and Open Problems*, New York, 1980. Academic Press, Inc.

[8] K. J. Gough. *Syntax Analysis and Software Tools.* International Computer Science Series. Addison Wesley Publishers, Sydney, 1988.

[9] J. Gruska. A characterization of context-free languages. *Journal of Computer and System Sciences*, 5:353 – 364, 1971.

[10] M. Harrison. *Introduction to Formal Languages.* Addison Wesley, Reading, Mass., 1978.

[11] J. Hopcroft and J. D. Ullman. *Formal Languages and their Relation to Automata.* Addison Wesley, Reading, Mass., 1979.

[12] D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. In *6th Annual Symposium on Logic in Computer Science, July 15–18, 1991, Amsterdam*, Los Alamitos, CA, 1991. Computer Society Press.

[13] J. Lambek. The mathematics of sentence structure. *American Mathematical Monthley*, 65:154–170, 1958.

[14] H. R. Lewis and C. H. Papadimitriou. *Elements of The Theory of Computation.* Prentice-Hall International, Inc., London, 1981.

[15] O. Mayer. *Syntaxanalyse.* Bibliographisches Institut, Mannheim, 1986.

[16] B. Partee, A. ter Meulen, and R. E. Wall. *Mathematical Methods in Linguistics.* Kluwer Academic Publishers, Dordrecht, 1990.

[17] M. Pentus. Lambek grammars are context free. In *Proc. 8th Annual IEEE Symposium on Logic in Computer Science, Montreal, Canada, June 19-23*, pages 429–433. IEEE Computer Society Press, 1993.

[18] H. Pfeiffer. *Oh Cello voll Echo. Palindrom-Gedichte*. Insel Verlag, Frankfurt/Main und Leipzig, 1992.

[19] J. E. Pin. *Varieties of Formal Languages*. North Oxford Academic Publishers, Kogan Page Ltd, 120 Pentonville Road, London N1 9JN, 1984.

[20] C. Pollard. *Generalized Phrase Structure Grammars, Head Grammars, and Natural Language*. PhD thesis, Department of Linguistics, Stanford University, 1984.

[21] V. Pratt. Action logic and pure induction. In *Logic in AI. Proceedings JELIA'90, Univ. of Amsterdam*, volume 478 of *Lecture Notes in Artificial Intelligence, Subseries of LNCS*, pages 97–120. Springer Verlag, 1991.

[22] V. N. Redko. On defining relations for the algebra of regular events (Russisch). *Ukrain. Mat. Z.*, 16:120–126, 1964.

[23] A. Salomaa. *Formal Languages*. Academic Press, Inc., New York and London, 1973.

[24] W. Savitch, editor. *The Formal Complexity of Natural Language*. Reidel, Dordrecht, 1987.

[25] W. Thomas. Classifying regular events in symbolic logic. *Journal of Computer and System Sciences*, 25:360–376, 1982.

[26] A. Thue. Probleme über Veränderungen von Zeichenreihen nach gegebenen Regeln. *Videnskapsselskapets Skrifter. I. Mat.-Naturv.Klasse.*, (10):3–34, Kristiania, 1914.

[27] A. Turing. On computable numbers with an application to the Entscheidungsproblem. In *London Mathematical Society*, volume 2-42, pages 230–265, 1936.

[28] W.W.Boone. The word problem. *Annals of Mathematics*, 70:207–265, 1959.