Formale Sprachen und Automaten

Klaus U. Schulz

13. Dezember 2011

Inhaltsverzeichnis

Kapitel 1

Grundlagen

1.1 Alphabete, Wörter und Sprachen

Definition 1.1 [Alphabet] Ein *Alphabet* ist eine endliche Menge Σ von Zeichen (Symbolen).

Es sei angemerkt, daß in anderen Kontexten auch unendliche Alphabete betrachtet werden. Diese sind aber nicht relevant für die nachfolgenden Darstellungen.

Beispiel 1.2 $\Sigma := \{0,1\}$, $\Sigma := \{a, \cdots, z\}$ sind Alphabete. Auch $\Sigma_0 = \{Mann, Frau, Kind, Der, Die, Den, Das, Lobt\}$ ist ein Alphabet (von Wörtern).

Für jedes $n \in \mathbb{N}$ bezeichne nachfolgend $S_n := \{i \in NN \mid 1 \leq i \leq n\}$ das Anfangsintervall der Länge n der natürlichen Zahlen. Hierbei ist S_0 die leere Menge $\emptyset$.

Definition 1.3 [Wort] Sei Σ ein Alphabet. Ein *Wort* der Länge $n \in NN$ über Σ ist eine Abbildung w: $S_n \to \Sigma$. Hierbei ist $w(i)$ das i-te Symbol von ω für $1 \leq i \leq n$.

Salopp: Ein Wort ist eine Folge von $n \geq 0$ Symbolen aus Σ.

Beispiel 1.4 Sei $\Sigma := \{a, b, c\}$. Dann ist $u = aab$ ein Wort über Σ. Formal kann u als die Abbildung mit dem Graphen $\{\langle 1, a\rangle, \langle 2, a\rangle, \langle 3, b\rangle\}$ beschrieben werden.

Beispiel 1.5 Es sei $\Sigma = \{Mann, Frau, Kind, Der, Die, Den, Das, Lobt\}$. „Das Kind Lobt Den Mann“ ist ein Wort über Σ; aber auch „Kind Frau Mann“.

Notation: Schreiben Wörter stets in der Form aab, nicht als Abbildung.

Bemerkung 1.6 Sei Σ beliebig. Das Wort der Länge 0 über Σ ist eindeutig bestimmt. Es ist die leere Abbildung $\emptyset$. Wir werden „ϵ“ als Symbol für das leere Wort verwenden.

Definition 1.7 Wir bezeichnen mit

- Σ^n die Menge aller Wörter der Länge n über Σ.
- $\Sigma^* = \bigcup_{i \in N} \Sigma^i$ die Menge aller Wörter über Σ,
- $\Sigma^+ = \bigcup_{i>0} \Sigma^i$ die Menge aller nichtleeren Wörter über Σ.

Definition 1.8 [Sprache] Eine (formale) *Sprache* über einem Alphabet Σ ist eine Teilmenge L von Σ^*.

Beispiel 1.9 $\{a, b\}^*$, $\{\epsilon\}$ und $\emptyset$ sind Sprachen über $\{a, b\}$.

Der nachfolgende Seitenausflug soll klarmachen, daß es für ein nichtleeres Alphabet Σ in einem zu präzisierenden Sinn mehr Sprachen über Σ gibt, als mit endlichen Ausdrucksmitteln in irgendeiner Weise darstellbar sind. Um diesen Sachverhalt zu präzisieren, benötigen wir einige Begriffe der Mengenlehre.

Definition 1.10 [abzählbar] Eine Menge M heißt *abzählbar* genau dann, wenn M endlich ist oder es eine Bijektion β: $N \to M$ gibt.

Lemma 1.11 *Sei Σ ein nichtleeres Alphabet. Dann ist Σ^* abzählbar unendlich.*

Beweis: Bleibt dem Leser überlassen. (Hinweis: mit Hilfe von Ordnungen.)

Bemerkung 1.12 Sei $\Sigma = \emptyset$. Dann ist $\Sigma^* = \{\epsilon\}$.

Lemma 1.13 *Ist M eine abzählbar unendliche Menge, so ist die Potenzmenge $\mathcal{P}(M)$ überabzählbar.*

Beweis: Zunächst ist klar, daß $\mathcal{P}(M)$ nicht endlich ist. Angenommen es gibt eine Bijektion $\beta : N \longrightarrow$P(M). Da M abzählbar unendlich ist, existiert eine Bijektion γ: $N \longrightarrow M$. Damit ist $\delta := \beta \circ \gamma^{-1}$ eine Bijektion $M \longrightarrow$P(M). Wir betrachten nun die Diagonalmenge $D := \{m \in M : \ m \notin \delta(m)\}$. Da δ bijektiv ist, existiert ein $n \in M$ mit $\delta(n) = D$. Falls

$$\begin{array}{l} n \in D, \ \ D = \delta(n), \text{ so gilt nach Definition } n \notin D \\ n \notin D, \ \ D = \delta(n), \text{ so gilt nach Definition } n \in D \end{array}$$

Dies ist ein Widerspruch, also kann es keine Bijektion β geben. ■

Bemerkung 1.14 Die obige Beweismethode, die ein „Diagonalargument“ verwendet, geht auf Georg Cantor, den Begründer der Mengenlehre, zurück. Man kann mit ähnlichen Diagonalargumenten z.B. zeigen, daß die reellen Zahlen überabzählbar sind.

Korollar 1.15 *Ist Σ ein nichtleeres Alphabet, so gibt es überabzählbar viele formale Sprachen über Σ.*

Das Lemma zeigt, daß es zu jedem nichtleeren Alphabet Σ eine überabzählbare Menge von Sprachen über Σ gibt. Wenn wir Sprachen in gleich welcher Form eindeutig beschreiben wollen, so sind wir auf eine Metasprache angewiesen, die für die Beschreibung verwendet wird. Jede Sprachbeschreibung sollte aber ein *endlicher* Ausdruck der Metasprache sein. Es ist sinnvoll anzunehmen, daß auch die Metasprache über einem endlichen nichtleeren Alphabet Γ gebildet ist. Sprachbescheibungen sind dann Wörter über Γ. Nach Lemma 1.11 kann es damit aber nur abzählbar unendlich viele Sprachbeschreibungen geben. Die Mehrzahl der Sprachen über Σ bleiben unbeschreibbar.

1.2 Operationen auf Wörtern

Definition 1.16 [Konkatenation] Die *Konkatenation* $\circ : \Sigma^* \times \Sigma^* \to \Sigma^*$ ist wie folgt definiert: für $u \in \Sigma^k$ und $v \in \Sigma^l$ ist $u \circ v \in \Sigma^{k+l}$ die Abbildung $S_{k+l} \to \Sigma$ mit

$$u \circ v(i) := \begin{cases} u(i) & \text{für} \quad 1 \leq i \leq k, \\ v(i-k) & \text{für} \quad k < i \leq k+l. \end{cases}$$

Beispiel 1.17 $aab \circ bc = aabbc$.

Definition 1.18 [Inversenbildung] Die *Inversenbildung* $^{-1}$: $\Sigma^* \to \Sigma^*$ ist wie folgt definiert: für $u \in \Sigma^l$ ist u^{-1}: $S_l \to \Sigma^l$ definiert als $u^{-1}(i) := u(l+1-i)$.

Beispiel 1.19 $aabc^{-1} = cbaa$.

Notation: statt u^{-1} wird in der Literatur auch oft u^{inv} oder u^{rev} geschrieben.

Bemerkung 1.20 a) Die Konkatenation „$\circ$“ ist assoziativ. Es ist $\langle \Sigma^+, \circ \rangle$ die von Σ frei erzeugte Halbgruppe und $\langle \Sigma^*, \circ, \epsilon \rangle$ das von Σ frei erzeugte Monoid mit dem neutralen Element ϵ.
b) Für jedes Wort $w \in \Sigma^*$ gilt: $(w^{-1})^{-1} = w$.

Notation: Anstatt $u \circ v \circ w$ schreiben wir kurz uvw.

Definition 1.21 [Teilwort, Präfix, Suffix] Seien $w_1, w_2 \in \Sigma^*$.

1. w_1 heißt *Teilwort* oder *Faktor* von w_2 genau dann, wenn es $u, v \in \Sigma^*$ gibt mit $w_2 = uw_1v$; das Paar (u, v) heißt dann der *Kontext* von w_1 in w_2.
2. w_1 heißt *Präfix* von w_2 genau dann, wenn es ein $u \in \Sigma^*$ gibt mit $w_2 = w_1u$.

3. w_1 heißt *Suffix* von w_2 genau dann, wenn es ein $u \in \Sigma^*$ gibt mit $w_2 = uw_1$.

Ein Teilwort (Präfix, Suffix) w_1 eines Worts w_2 heißt *echtes* Teilwort (Präfix, Suffix) von w_2 genau dann, wenn $w_1 \neq w_2$ gilt.

Beispiel 1.22 1. 01 ist Teilwort, Suffix und Präfix von 01.

2. 0 ist *echtes Präfix* von 01.

3. 1 ist *echtes Suffix* von 01.

4. ϵ ist Teilwort, Präfix und Suffix jedes Wortes.

1.3 Operationen auf Sprachen

Wir werden im weiteren Verlauf die folgenden Operationen auf Sprachen über einem festen Alphabet Σ verwenden (es bezeichnen L, L_1 und L_2 Sprachen über Σ).

- *Vereinigung* $L_1 \cup L_2$,
- *Durchschnitt* $L_1 \cap L_2$,
- *Differenz* $L_1 - L_2$,
- *Komplement* $\bar{L} := \Sigma^* \setminus L$,
- *Konkatenation* $L_1 \circ L_2 := \{w_1 w_2 \mid w_1 \in L_1, w_2 \in L_2\}$,
- n-te *Potenz* $L^n := \{w_1 \cdots w_n \mid w_i \in L \text{ für } 1 \leq i \leq n\}$ $(n \geq 1)$, sowie $L^0 := \{\epsilon\}$,
- *Konkatenationsabschluß* oder *Kleene-Stern* $L^* := \bigcup_{n \in N} L^n$.
- *Linksdivision* von L_2 durch L_1 als

$$L_1 \setminus L_2 := \{w \in \Sigma^* \mid \forall v \in L_1 : v \circ w \in L_2\},$$

- *Rechtsdivision* von L_1 durch L_2 als

$$L_1 / L_2 := \{w \in \Sigma^* \mid \forall v \in L_2 : w \circ v \in L_1\},$$

- *Spiegelung* $L^{-1} := \{w^{-1} \mid w \in L\}$.

Einige Bemerkungen:

1. Falls $L_1 = \emptyset$ oder $L_2 = \emptyset$, so folgt $L_1 \circ L_2 = \emptyset$.
2. Es gilt $L^n = L \circ L \cdots \circ L$ (n Faktoren).
3. Der Name Links- (bzw. Rechts-) Division wird verständlicher mit folgenden Regeln. Stets gilt $L_1 \circ (L_1 \setminus L_2) \subseteq L_2$ und $(L_1/L_2) \circ L_2 \subseteq L_1$.

Beispiel 1.23

- $\emptyset^* = \{\epsilon\}$
- $\{\epsilon\}^* = \{\epsilon\}$
- $L^* = L^+ \cup \{\epsilon\}$, *wobei* $L^+ = \bigcup_{i \in N \setminus \{0\}} L^i$.

1.4 Drei Grundprobleme

Zum Abschluss des Kapitels wollen wir drei Grundproblemen nennen, die sich beim Umgang mit formalen Sprachen immer wieder stellen.

1. *Beschreibungs-* bzw. *Definitionsproblem.* Welche Formalismen lassen sich verwenden, um formale Sprachen eindeutig zu beschreiben bzw. zu definieren.
2. *Erkennungs-* bzw. *Entscheidungsproblem.* Gegeben sei eine formale Sprache L über einem Alphabet Σ, die in geeigneter Weise beschrieben sei. Wie kann man für vorgelegte Wörter $w \in \Sigma^*$ entscheiden, ob $w \in L$ gilt?
3. *Analyseproblem.* Gegeben sei eine formale Sprache L über einem Alphabet Σ, die derart beschrieben sei, daß jedes Wort $w \in L$ eine (nicht notwendig eindeutige) grammatikalische Struktur besitze. Wie kann man für ein Eingabewort $w \in \Sigma^*$ entscheiden, ob $w \in \Sigma^*$ gilt *und* gegebenenfalls die grammatikalische Struktur(en) von w ermitteln?

Das Beschreibungsproblem führt uns auf verschiedene Grammatikbegriffe. Im weiteren Verlauf werden wir unter anderem reguläre Ausdrücke und kontextfreie Grammatiken als Beispiele kennelernen. Das Erkennungsproblem motiviert die Einführung verschiedener Typen von Automaten. Das Analyseproblem, welches das Erkennungsproblem beinhaltet, führt auf verschiedene Parsingverfahren. Die genannten Begriffe werden in den nachfolgenden Kapiteln präzisiert.

Aufgaben zu Kapitel 1

Aufgabe 1.1 Es seien u, v, w nichtleere Worte über einem Alphabet, die die „Konjugationsgleichung“ $uv = vw$ erfüllen. Was läßt sich über die Form von u bzw. den Zusammenhang zwischen der Gestalt von u und v sagen? Versuchen Sie, die allgemeine Gestalt von u in Abhängigkeit von v zu charakterisieren. Anleitung: betrachten Sie Situationen, wo v sehr viel länger ist als u.

Aufgabe 1.2 Es sei Σ das Alphabet $\{a, b\}$. Für $w \in \Sigma^*$ bezeichne w^i die i-fache Konkatenation $w \circ \cdots \circ w$ $(i \geq 1)$.

1. Es sei $L := \{a^i \mid 1 \leq i \leq k\}$, Was ist L^n?
2. Es sei $L := \{a\}$. Was ist $(L^2)^*$?
3. Was ist $(\Sigma \circ \Sigma)^*$? (Symbole aus Σ werden hier als Wörter der Länge 1 betrachtet.)

Aufgabe 1.3 Es sei L eine Sprache über dem Alphabet Σ. Wann gilt $L^+ = L^*$? Geben Sie eine notwendige und hinreichende Bedingung an.

Aufgabe 1.4 Es seien L, L_1 und L_2 Sprachen über dem Alphabet Σ. Zeigen Sie, daß stets gilt:

$$\begin{aligned}(L_1 \cup L_2)^{-1} &= L_1^{-1} \cup L_2^{-1},\\ (L_1 \cap L_2)^{-1} &= L_1^{-1} \cap L_2^{-1},\\ (L_1 \circ L_2)^{-1} &= L_2^{-1} \circ L_1^{-1},\\ (L^{-1})^{-1} &= L,\end{aligned}$$

$$\begin{aligned}(L_1 \cup L_2)^* &= (L_1^* \cup L_2^*)^*, \\ L \circ (L_1 \cup L_2) &= (L \circ L_1) \cup (L \circ L_2).\end{aligned}$$

Zeigen Sie, daß hingegen i.a. nicht $L \circ (L_1 \cap L_2) = (L \circ L_1) \cap (L \circ L_2)$ gilt.

Kapitel 2

Reguläre Sprachen und endliche Automaten

2.1 Reguläre Ausdrücke und reguläre Sprachen

Reguläre Ausdrücke stellen eine wichtige Möglichkeit dar, möglicherweise unendliche Sprachen mit Hilfe endlicher Metaausdrücke zu definieren. Sie stellen damit *eine* partielle Lösung des im vorausgegangenen Kapitel genannten „Beschreibungsproblems“ dar. Es bezeichne nachfolgend Σ stets ein endliches Alphabet.

Definition 2.1 [reguläre Ausdrücke] Die Menge der *regulären Ausdrücke* über Σ ist rekursiv wie folgt definiert:

1. $\emptyset$ (alternativ auch als 0 notiert) ist ein regulärer Ausdruck.
2. 1 ist ein regulärer Ausdruck.
3. Für jedes $\sigma \in \Sigma$ ist σ ein regulärer Ausdruck.
4. Falls α und β reguläre Ausdrücke sind, so auch
   - α^*,
   - $(\alpha \cup \beta)$ (alternativ auch als $(\alpha + \beta)$ notiert) und

- $(\alpha \cdot \beta)$ (alternativ auch als $(\alpha\beta)$ notiert).

5. Jeder reguläre Ausdruck wird mit den oberen vier Regeln gebildet.

Es ist wichtig, sich klarzumachen, daß die in den regulären Ausdrücken verwendeten Symbole wie $\emptyset$, $\cup$, σ, hier Ausdrücke einer *Metasprache* sind. Beispielsweise macht es keinen Sinn, von der Vereinigung (im mengentheoretischen Sinn) zweier regulärer Ausdrücke zu sprechen. Reguläre Ausdrücke sind demzufolge keine Sprachen, sie werden vielmehr verwendet, um Sprachen zu bezeichnen.

Definition 2.2 Jedem regulären Ausdruck α über Σ läßt sich rekursiv eine Sprache $L(\alpha) \subseteq \Sigma^*$ wie folgt zuordnen:

1. $L(\emptyset) = \emptyset$ $(= L(0))$,
2. $L(1) = \{\epsilon\}$,
3. $L(\sigma) = \{\sigma\}$ für alle $\sigma \in \Sigma$,
4. $L(\alpha^*) = L(\alpha)^*$,
5. $L((\alpha \cup \beta)) = L(\alpha) \cup L(\beta)$ $(= L(\alpha + \beta))$,
6. $L((\alpha \cdot \beta) = L(\alpha) \circ L(\beta)$ $(= L(\alpha\beta))$.

Es folgt aus Definition 2.2, daß der Ausdruck 1 redundant ist. In der Tat könnten wir ihn durch 0^* ersetzen. Manche Autoren verzichten daher auf diesen Ausdruck.

Beispiel 2.3 Sei $\Sigma = \{0, 1\}$.
Es ist $L((1^* \cup 0^*)) = L(1^*) \cup L(0^*) = L(1)^* \cup L(0)^* = \{1\}^* \cup \{0\}^*$.

Wir werden in regulären Ausdrücken nachfolgend Klammern weglassen, wenn die eindeutige Lesbarkeit erhalten bleibt.

Definition 2.4 Die Menge der *regulären Sprachen* über Σ ist die kleinste Menge von Sprachen, die die Sprachen $\emptyset$ sowie $\{\sigma\}$ für alle $\sigma \in \Sigma$ enthält, und die abgeschlossen ist unter den „regulären“ Operationen $\circ$, * und $\cup$.

Bemerkung 2.5 1. Eine solche Menge existiert, da mit jeder Menge $\mathcal{L}$ von Sprachfamilien, die alle die obigen Eigenschaften besitzen, stets auch deren Durchschnitt $\bigcap \mathcal{L}$ eine Sprachfamilie ist, die alle aufgeführten Eigenschaften besitzt.

2. Es ist leicht zu zeigen: $L \subseteq \Sigma^*$ ist eine reguläre Sprache genau dann, wenn ein regulärer Ausdruck α über Σ existiert mit $L = L(\alpha)$.

3. Jede endliche Sprache über Σ ist stets regulär.

4. Wir werden später sehen, daß die Klasse der regulären Sprachen auch unter Durchschnitt $\cap$ und Differenz $-$ abgeschlossen ist.

Definition 2.6 Zwei reguläre Ausdrücke α und α' heißen *äquivalent* genau dann, wenn $L(\alpha) = L(\alpha')$ gilt.

Wir schreiben $\alpha \equiv \beta$ falls die regulären Ausdrücke α und β äquivalent sind. Reguläre Ausdrücke werden in vielen Programmiersprachen zum „Patternmatching“ (Suche nach Ausdrücken, die einem einfachen Muster folgen) in Text- oder Programmdateien verwendet. Auch in Betriebssystemen werden — etwa zur Dateiensuche — verwandte Ausdrücke wie $*.dvi$ verwendet. Die resultierenden Suchzeiten können für zwei äquivalente reguläre Ausdrücke sehr unterschiedlich sein. Für Optimierungszwecke und verwandte Aufgaben ist man also daran interessiert, einfache Regeln zur Äquivalenzumformung zur Verfügung zu haben. Das folgende Lemma listet einige wichtige Regeln auf.

Lemma 2.7 *Seien α, β und γ reguläre Ausdrücke. Dann gilt stets:*

1. $1 \cdot \alpha \equiv \alpha \cdot 1 \equiv \alpha$,
2. $\emptyset \cdot \alpha \equiv \alpha \cdot \emptyset \equiv \emptyset$,
3. $\alpha \cup \beta \equiv \beta \cup \alpha$,
4. $(\alpha \cup \beta) \cdot \gamma \equiv (\alpha \cdot \gamma) \cup (\beta \cdot \gamma)$ *und*
   $\gamma \cdot (\alpha \cup \beta) \equiv (\gamma \cdot \alpha) \cup (\gamma \cdot \beta)$,
5. $\alpha^* \equiv \emptyset^* \cup \alpha\alpha^*$,
6. $\emptyset^*\alpha \equiv \alpha$

7. *Falls* $\alpha \equiv \beta^*\gamma$ *so* $\alpha \equiv \beta\alpha \cup \gamma$

8. *Falls* $\epsilon \notin L(\beta)$ *und* $\alpha \equiv \beta\alpha \cup \gamma$ *so auch* $\alpha \equiv \beta^*\gamma$.

Beweis: Der Nachweis der Äquivalenzen 1-6 ist einfach und wird als Übungsaufgabe dem Leser überlassen.

Zu 7. $\alpha \equiv \beta^*\gamma \equiv (\beta\beta^* \cup \emptyset^*)\gamma \equiv \beta\beta^*\gamma \cup \gamma \equiv \beta\alpha \cup \gamma$.

Zu 8. Hilfsrechnung:

$$\begin{aligned} & \alpha \\ \equiv\ & \beta\alpha \cup \gamma \equiv \beta(\beta\alpha \cup \gamma) \cup \gamma \\ \equiv\ & \beta\beta\alpha \cup (\beta \cup \emptyset^*)\gamma \equiv \beta\beta(\beta\alpha \cup \gamma) \cup (\beta \cup \emptyset^*)\gamma \\ \equiv\ & \beta\beta\beta\alpha \cup (\beta\beta \cup \beta \cup \emptyset^*)\gamma \\ \equiv\ & \cdots \\ \equiv\ & \beta^{k+1}\alpha \cup (\beta^k \cup \beta^{k-1} \cup \cdots \cup \beta \cup \emptyset^*)\gamma \end{aligned}$$

„$\subseteq$“: Sei $w \in L(\alpha)$, $\mid w \mid =: k$. Jedes $v \in L(\beta^{k+1}\alpha)$ *hat* $\mid v \mid \geq k+1$. Daher existiert ein $i \leq k:\ w \in L(\beta^i\gamma) \subset L(\beta^*\gamma)$.
„$\supseteq$“: Sei $w \in L(\beta^*\gamma)$. Dann gibt es ein i so, daß $w \in L(\beta^i\gamma)$, $w \in L(\alpha)$. ■

Bemerkung 2.8 Operationen $\cup$, $\circ$ und * mit ähnlichen Gesetzmäßigkeiten treten auch bei der Menge der zweistelligen Relationen über einer Menge M auf (Vereinigung, Komposition und reflexiv-transitive Hülle). Eine alternative Interpretation regulärer Ausdrücke als binäre Relationen macht daher Sinn. Weiterhin können reguläre Ausdrücke auch in einer dritten Form als Programme interpretiert werden. Zu den Einzelheiten der beiden alternativen Interpretationen vergleiche man [**?**].

2.2 Deterministische endliche Automaten

Die sogenannten endlichen Automaten – wovon die deterministischen endlichen Automaten eine Teilklasse bilden — stellen einen einfachen Typ von abstrakten Maschinen dar, mit denen das in Abschnitt 1.4 beschriebene „Erkennungsproblem“ gelöst werden kann, allerdings nur für die Klasse der

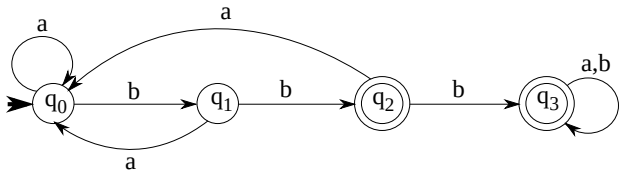


Abbildung 2.1: Deterministischer endlicher Automat in Graphendarstellung

regulären Sprachen.[1] Jeder endliche Automat definiert in natürlicher Weise eine Sprache und einen zugehörigen *Erkennungsalgorithmus*. Die Eingabe des Algorithmus ist ein beliebiges Wort über Σ. Das Ergebnis ist die Ausgabe „ja“ genau dann, wenn das Eingabewort zu der dem Automaten zugehörigen Sprache gehört. Wir werden später sehen, daß man mit Hilfe endlicher Automaten *genau* die regulären Sprachen erkennen kann.

Bevor wir deterministische endliche Automaten und deren Erkennungsprozedur formal definieren, geben wir in Abbildung 2.1 ein einfaches Beispiel. Aus Gründen der Übersichtlichkeit stellen wir den Beispielautomaten als einen gerichteten Graphen dar. Der abgebildete Automat hat vier *Zustände* $q_0, \ldots, q_3$, die den Knoten des Graphen entsprechen. Zustand q_0 ist als sogenannter *Startzustand* ausgezeichnet und entsprechend mit einem Eingangspfeil markiert. Zwei der Zustände, nämlich q_2 und q_3 sind als sogenannte *Finalzustände* ausgezeichnet und entsprechend durch eine zweite Umrandung markiert. Die mit Symbolen des *Eingabe-Alphabets* $\Sigma := \{a, b\}$ gelabelten Kanten werden als die *Übergänge* des Automaten bezeichnet. Man kann a-Übergänge und b-Übergänge unterscheiden. Für die hier abgebildeten *deterministischen* endlichen Automaten charakteristisch ist, daß man von jedem Zustand und für jedes Symbol σ des Eingabe-Alphabets genau eine ausgehende Kante bzw. einen σ-Übergang hat.

Die durch den Automaten definierte Erkennungsprozedur ist nun sehr einfach zu erklären. Wir fassen ein zu testendes Wort $w = \sigma_1 \ldots \sigma_n$ als „Ticketfolge“ auf, die Kantenlabels als zu bezahlende Tickets. Ein Wort wird genau dann akzeptiert, wenn man vom Startzustand aus startend durch

[1] Nicht für jede Sprache ist das Problem überhaupt lösbar. Wenn es überhaupt lösbar ist, so noch nicht notwendigerweise mit Hilfe eines endlichen Automaten

Traversieren der durch die Tickets $\sigma_1, \ldots, \sigma_n$ des Eingabeworts vorgegebenen Übergänge nach Verbrauch aller Tickets in einem Finalzustand endet. So wird das Eingabewort *babb* vom Automaten akzeptiert, da die vier ,,Tickets" b, a, a, b des Wortes vom Startzustand beginnend zunächst zu q_1, dann zu q_0, dann zu q_1 und schließlich zum Finalzustand q_2 führen. Das Wort *bab* hingegen wird verworfen, da man vom Startzustand aus mit der Ticketfolge b, a, b nach dem nicht-finalen Zustand q_1 gelangt. Wir wollen diese Idee nun formalisieren.

Bemerkung 2.9 Die Begriffe des ,,Zustands" und des ,,Übergangs" sind im Zusammenhang mit Automaten von zentraler Bedeutung. Unter einem Zustand sollte man sich intuitiv allerdings weniger den Knoten eines Graphen vorstellen, als eine Momentaufnahme eines dynamischen Systems zu einem festen Zeitpunkt. Dabei wird angenommen, daß das System in jedem Moment vollständig durch eine endliche Zahl von Parameterwerten (aus einem endlichen Vorrat möglicher Werte) beschreibbar ist. Übergänge werden durch äußere Impulse ausgelöst, sie führen in einen Nachfolgerzustand. Bei deterministischen endlichen Automaten bestimmt jeder Impuls den Nachfolgerzustand in eindeutiger Weise.

Definition 2.10 [DEA] Ein *deterministischer endlicher Automat* (DEA) ist ein Quintupel $A = (Q, \Sigma, \delta, s, F)$ wo gilt:

- Q ist eine endliche Menge, genannt die Menge der *Zustände* von A,
- Σ ist ein endliches Alphabet, das *Eingabe-Alphabet*,
- $\delta : Q \times \Sigma \to Q$ ist eine Funktion, genannt die *Übergangsfunktion*,
- $s \in Q$ ist ein ausgezeichneter *Startzustand*,
- $F \subseteq Q$ ist eine ausgezeichnete Menge von *Finalzuständen*

Um die Bearbeitung einer Eingabe durch einen endlichen Automaten zu formalisieren, ist der Begriff der Konfiguration nützlich.

Definition 2.11 [Übergang] Eine *Konfiguration* von A ist ein Element aus $Q \times \Sigma^*$ (lies: ein Paar bestehend aus dem aktuellem Zustand und dem

Restwort). Der DEA A erlaubt den *direkten Übergang* von der Konfiguration (q, w) zur *Nachfolgerkonfiguration* (q', w'), im Zeichen

$$(q, w) \Longrightarrow_A^1 (q', w')$$

genau dann, wenn w die Form $\sigma w'$ $(\sigma \in \Sigma)$ hat, wo $\delta(q, \sigma) = q'$. Mit $\Longrightarrow_A^*$ wird die reflexive und transitive Hülle der Relation $\Longrightarrow_A^1$ bezeichnet. Das Wort $w \in \Sigma^*$ wird von A *akzeptiert* genau dann, wenn es einen Finalzustand $f \in F$ gibt mit

$$(s, w) \Longrightarrow_A^* (f, \epsilon).$$

Die Sprache $L(A) := \{w \in \Sigma^* \mid A \text{ akzeptiert } w\}$ heißt die *von A erkannte Sprache*, oder die Sprache von A.

Eine alternative Art und Weise, die Sprache $L(A)$ zu charakterisieren, ergibt sich durch die Einführung der *verallgemeinerten Übergangsfunktion* $\hat{\delta} : Q \times \Sigma^* \to Q$. Diese ist induktiv wie folgt erklärt.

$$\begin{aligned} &\hat{\delta}(q, \epsilon) := q, \\ &\hat{\delta}(q, w\sigma) := \delta(\hat{\delta}(q, w), \sigma) \quad (w \in \Sigma^*, \sigma \in \Sigma). \end{aligned}$$

Lemma 2.12 *Sei A ein DEA wie oben. Dann gilt stets für jedes $w \in \Sigma^*$ und alle $p, q \in Q$: $(p, w) \Longrightarrow_A^* (q, \epsilon)$ genau dann, wenn $\hat{\delta}(p, w) = q$.*

Beweis. Durch Induktion über $|w|$.

Für $|w| = 0$ gilt $w = \epsilon$. Wir haben $(p, \epsilon) \Longrightarrow_A^* (q, \epsilon)$ gdw. $p = q$ gdw. $\hat{\delta}(p, \epsilon) = q$.

Sei die Behauptung bewiesen für alle Wörter der Länge $n \geq 0$. Sei $w = w'\sigma$ ein Wort der Länge $n + 1$ über Σ und $\sigma \in \Sigma$. Wir haben $(p, w'\sigma) \Longrightarrow_A^* (q, \epsilon)$ gdw. es existiert $r \in Q : (p, w'\sigma) \Longrightarrow_A^* (r, \sigma) \Longrightarrow_A^1 (p, \epsilon)$ gdw. es existiert $r \in Q : \hat{\delta}(p, w') = r$ und $\delta(r, \sigma) = q$ gdw. $\hat{\delta}(p, w'\sigma) = q$. Dies beendet den Beweis. □

Korollar 2.13 *Unter Verwendung der obigen Bezeichnungen gilt $L(A) = \{w \in \Sigma^* \mid \hat{\delta}(s, w) \in F\}$.*

Beweis. Aus dem Lemma ergibt sich für beliebiges $w \in \Sigma^*$: $w \in L(A)$ gdw. $(s, w) \Longrightarrow_A^* (f, \epsilon)$ für ein $f \in F$ gdw. $\hat{\delta}(s, w) = f$ mit $f \in F$. □

Von dieser Charakterisierung wird an verschiedener Stelle Gebrauch gemacht. Wir werden sehen, daß man zu einem gegebenen regulären Ausdruck α stets einen DEA A konstruieren kann mit $L(A) = L(\alpha)$ und vice versa. Als Beispiel zeigt Abbildung 2.2 einen DEA A, der die Sprache $L((ab \cup aab)b^*(a(ab \cup aab)b^*)^*)$ erkennt.

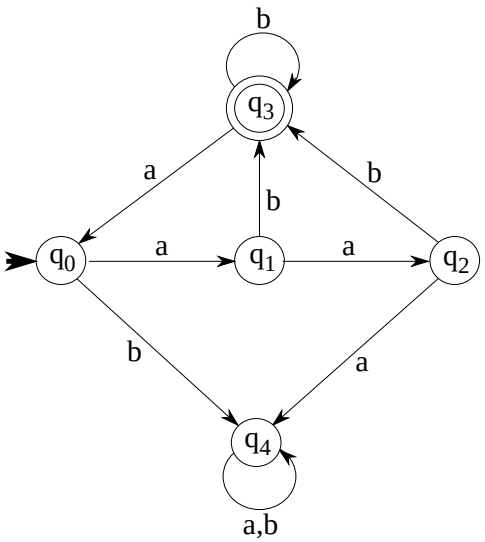


Abbildung 2.2: DEA für $(ab \cup aab)b^*(a(ab \cup aab)b^*)^*$

Eine von vielen Anwendungen von deterministischen endlichen Automaten im Bereich der Computerlinguistik illustriert Abbildung 2.3: Man kann auch extrem umfangreiche Wörterbücher in sehr kompakter Weise als deterministische endliche Automaten abspeichern. Auf diese Art kann extrem schnell festgestellt werden, ob ein gegebenes Eingabewort im Lexikon ist.

Bemerkung 2.14 Die Übergangsfunktion eines deterministischen endlichen Automaten kann als Tabelle bzw. Matrix M der Dimension $|Q| \times |\Sigma|$ dargestellt werden. Der Eintrag $M_{i,j}$ gibt das Bild des i-ten Zustands unter dem j-ten Symbol des Alphabets an. Für den in Abbildung 2.1 beschriebenen Automaten erhält man beispielsweise die folgende Tabelle:

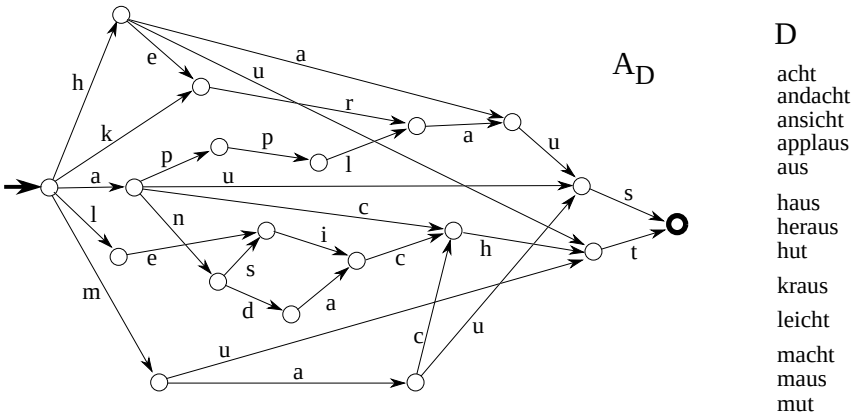


Abbildung 2.3: Wörterbuch D und Kodierung als deterministischer endlicher Automat A_D.

	a	b
q_0	q_0	q_1
q_1	q_0	q_2
q_2	q_0	q_3
q_3	q_3	q_3

Bei der Implementierung deterministischer endlicher Automaten werden oft Tabellen dieser Form verwendet. Sind nicht alle Übergänge realisiert, so kann man platzsparender auch jedem Zustand eine Liste der von ihm weggehenden Übergänge zuordnen.

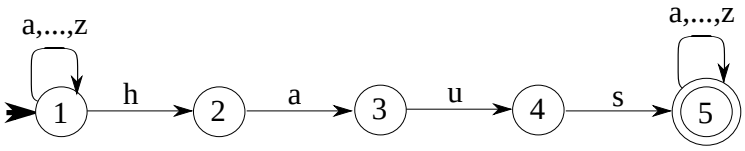
Bemerkung 2.15 Die Bedeutung der *deterministischen* endlichen Automaten beruht vor allem darauf, daß sie für das eingangs beschriebene Erkennungsproblem einen Algorithmus liefern, dessen Zeitkomplexität *linear* in der Länge des Eingabeworts ist. Ist $L = L(A)$ die Sprache des deterministischen endlichen Automaten A und ist w ein Eingabewort der Länge n, so kann man mittels n Aufrufen der Übergangsfunktion und einem anschließenden Test auf Finalität des erreichten Zustands überprüfen, ob $w \in L$ gilt. Ein Aufruf der Übergangsfunktion kann in konstanter Zeit realisiert werden, etwa indem man die Übergangsfunktion wie oben dargestellt als Tabelle implementiert.

Bemerkung 2.16 Wie die vorausgegangene Bemerkung deutlich macht, ist es für das Konzept des deterministischen endlichen Automaten nicht wesentlich, daß es von jedem Zustand für *jedes* Symbol σ des Alphabets einen weglaufenden Übergang mit Label σ gibt. Solange wir pro Zustand q für jedes $\sigma \in \Sigma$ stets *höchstens* einen von q startenden Übergang mit Label σ haben, und solange jeder Übergang als Label ein Wort der Länge 1 hat, ergeben sich unveränderte Erkennungsmöglichkeiten und Komplexitäten. Wir erläutern dies anhand Abbildung 2.2. Beim dort abgebildeten Automaten könnten wir ohne Schaden alle zum Zustand q_4 führenden Übergänge weglassen, da wir von q_4 nicht mehr in einen Finalzustand gelangen können. Bei der Erkennungsprozedur würden wir das Eingabewort verwerfen, sobald wir eine Konfiguration $(q, \sigma w)$ erreichen, wo es von q aus keinen σ-Übergang mehr gibt. Wenn wir uns umgekehrt einen Automaten vorstellen, wo es Zustände gibt, die nicht für alle Symbole des Eingabe-Alphabets einen entsprechenden Übergang haben, so können wir den Automaten um einen neuen sogenannten ,,Fallenzustand“ ergänzen. Dieser ist nicht-final, alle fehlenden Übergange lassen wir im Fallenzustand enden. Vom Fallenzustand führt jedes Symbol stets wieder zum Fallenzustand.

2.3 Nicht-deterministische endliche Automaten

Die nun zu besprechenden nicht-deterministischen endlichen Automaten unterscheiden sich von den deterministischen endlichen Automaten vor allem dadurch, daß eine Konfiguration gleich *mehrere* mögliche Nachfolgerkonfigurationen haben kann. Dies macht es in der Regel schwieriger, die Akzeptanz eines Eingabeworts durch einen vorgegebenen Automaten algorithmisch zu überprüfen, wie wir sehen werden. Der Vorteil der nichtdeterministischen endlichen Automaten beruht darauf, dass die direkte Konstruktion eines deterministischen Automaten für eine gegebene reguläre Sprache oft sehr schwierig ist, die Konstruktion eines nichtdeterministischen endlichen Automaten hingegen in der Regel einfach.

Beispiel 2.17 Es soll ein endlicher Automat konstruiert werden, der alle Zeichenfolgen erkennt, die das Teilwort ,,haus“ enthalten. Die nachfolgende Abbildung zeigt einen geeigneten nicht-deterministischen endlichen Automaten A. Der Nichtdeterminismus beruht darauf, das im Zustand 1 zwei Übergänge mit Symbol “h” existieren.



Die direkte Konstruktion eines deterministischen endlichen Automaten ist sehr viel schwieriger.

Der genannte Vorteil nichtdeterministischer endlicher Automaten wird später noch deutlicher werden, wenn wir zeigen, wie man in systematischer Weise zu einem vorgegebenen regulären Ausdruck einen nichtdeterministischen endlichen Automaten konstruieren kann.

Definition 2.18 [NDEA] Ein *nicht-deterministischer endlicher Automat* (NDEA) A ist ein Quintupel $A = (Q, \Sigma, \Delta, s, F)$ wobei Zustandsmenge Q, Eingabe-Alphabet Σ, Startzustand $s \in Q$ und die Menge der Finalzustände $F \subseteq Q$ wie bei deterministischen endlichen Automaten erklärt sind. Es ist $\Delta \subseteq Q \times \Sigma^* \times Q$ eine Relation, die *Übergangsrelation* von A.

Statt von NDEA spricht man manchmal auch einfacher von endlichen Automaten, da jeder DEA auch als NDEA aufgefasst werden kann. Die Elemente der Menge Δ werden Übergänge von A genannt. Elemente der Form (p, ϵ, q) heißen *leere Übergänge*.

Definition 2.19 Eine *Konfiguration* eines NDEA A ist ein Paar aus $Q \times \Sigma^*$. A erlaubt den *direkten Übergang* von (q, w) zu (q', w'), im Zeichen $(q, w) \Longrightarrow_A^1 (q', w')$ genau dann, wenn w die Form uw' für ein $u \in \Sigma^*$ hat, wo $(q, u, q') \in \Delta$. Mit $\Longrightarrow_A^*$ bezeichnen wir wie üblich die reflexive transitive Hülle von $\Longrightarrow_A^1$. Das Wort $w \in \Sigma^*$ wird von A genau dann *akzeptiert*, wenn es ein $f \in F$ gibt mit $(s, w) \Longrightarrow_A^* (f, \epsilon)$. Die *von A akzeptierte Sprache* ist $L(A) := \{w \in \Sigma^*$: A akzeptiert w$\}$.

Definition 2.20 Sei $A = (Q, \Sigma, \Delta, s, F)$ ein NDEA. Die *verallgemeinerte Übergangsrelation* $\hat{\Delta} \subseteq Q \times \Sigma^* \times Q$ ist die kleinste Menge, für die gilt:

1. Für jedes $q \in Q$ ist $(q, \epsilon, q) \in \hat{\Delta}$.

2. Mit $(q_1, w_1, q_2) \in \hat{\Delta}$ und $(q_2, w_2, q_3) \in \Delta$ ist auch $(q_1, w_1 w_2, q_3) \in \hat{\Delta}$.

Bildlich gesprochen gehören zu $\hat{\Delta}$ genau diejenigen Tripel (q_1, w, q_2), wo man vom Zustand q_1 unter Konsum der Teileingabe w auf irgendeinem Weg nach q_2 gelangen kann. Mit diesem Begriff gelangen wir zu einer alternativen Beschreibung der Sprache eines NDEA. Es ist $L(A) = \{w \in \Sigma^* \mid \exists f \in F : (s, w, f) \in \hat{\Delta}\}$.

Wir wollen nun genauer betrachten, wie ein man die Akzeptanz eines Eingabeworts durch einen nichtdeterministischen endlichen Automaten algorithmisch überprüfen kann. Hierzu macht es Sinn, drei Typen nichtdeterministischer endlicher Automaten zu unterscheiden:

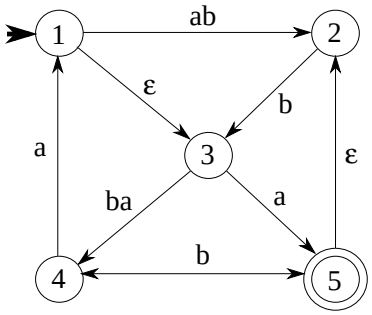
1. *Typ 1.* Beliebige NDEAs wie oben eingeführt.

2. *Typ 2.* NDEAs wo jede Kante mit einem Wort der Länge ≤ 1 gelabelt ist.

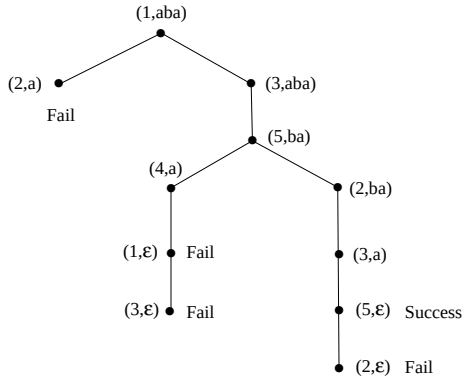
3. *Typ 3.* NDEAs wo jede Kante mit einem Symbol aus dem Eingabealphabet gelabelt ist.

1. Akzeptanzüberprüfung durch Backtracking. Eine erste Möglichkeit beruht darauf, in jeder Situation, wo sich mehrere mögliche Nachfolgerkonfigurationen ergeben, zunächst eine dieser Konfigurationen auszuwählen, die dann weiterverfolgt wird. Die Alternativen werden jedoch gemerkt, im Falle eines Scheiterns werden diese nacheinander ausprobiert. Man kann die Gesamtheit der so entstehenden Suchpfade durch einen endlich verzweigenden Baum darstellen. Die Methode funktioniert für alle drei Typen nichtdeterministischer endlicher Automaten, solange es keine Schleifen von leeren Übergängen gibt. Das nachfolgende Beispiel illustriert das Vorgehen.

Beispiel 2.21 Die nachfolgende Abbildung zeigt einen nichtdeterministischen endlichen Automaten A vom Typ 3 mit den Zuständen $1, \ldots, 5$.



Die Konfigurationsfolgen, die sich bei Eingabe des Worts *aba* ergeben, sind im nachfolgenden Suchbaum dargestellt. Bei einer Implementierung werden die Pfade des Suchbaums nacheinander durchlaufen.



Um eine alternative Möglichkeit der Akzeptanzüberprüfung darzustellen, ist der nachfolgende Begriff nützlich.

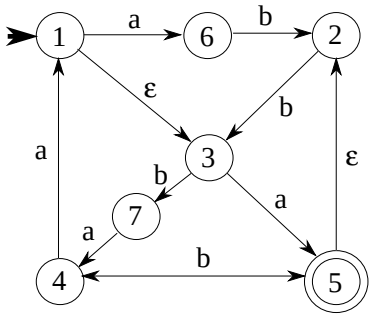
Definition 2.22 Sei $A = (Q, \Sigma, \Delta, s, F)$ ein NDEA und $w \in \Sigma^*$. Wir sagen, daß der Zustand $q \in Q$ bei Eingabe w *aktiv* ist, genau dann, wenn $(s, w, q) \in \hat{\Delta}$ gilt. Mit $Akt_A(w)$ bezeichnen wir die Menge der bei Eingabe w aktiven Zustände.

.

2. Simultane Berechnung der Menge der aktiven Zustände. Anstatt Backtracking zu verwenden, kann man auch simultan für jedes Präfix v des Eingabeworts w die Menge aller aktiven Zustände $Akt_A(v)$ berechnen und am

Ende prüfen, ob sich hierunter ein Finalzustand befindet. Die Berechnung der jeweiligen Menge der aktiven Zustände ist zwar prinzipiell für jeden NDEA möglich. Einfacher wird sie jedoch, wenn wir uns auf NDEA vom Typ 2 beschränken. In diesem Fall lassen sich Mengen der Form $Akt_A(v\sigma)$ $(v \in \Sigma^*, \sigma \in \Sigma)$ in einfacher Weise aus den Vorgängermengen $Akt_A(v)$ berechnen. Wir illustrieren dies mit einer Variante des vorausgegangenen Beispiels.

Beispiel 2.23 Die nachfolgende Abbildung zeigt einen nichtdeterministischen endlichen Automaten A vom Typ 2 mit den Zuständen $1, \ldots, 7$.



Die induktive Berechnung der Menge der aktiven Zustände bei Eingabe aba ist wie folgt:

1. Er ist $Akt_A(\epsilon) = \{1, 3\}$ die Menge der Zustände, die sich vom Startzustand aus durch $k \geq 0$ ϵ-Übergänge erreichen lassen.
2. $Akt_A(a) = \{6, 5, 2\}$ ergibt sich daraus, daß wir auf die Zustände $1, 3$ aus $Akt_A(\epsilon)$ (wo möglich)
   - einen a-Übergang,
   - gefolgt von $k \geq 0$ ϵ-Übergängen

   anwenden.
3. $Akt_A(ab) = \{3, 4, 2\}$ ergibt sich daraus, daß wir auf die Zustände $2, 5, 6$ aus $Akt_A(a)$ (wo möglich)
   - einen b-Übergang,
   - gefolgt von $k \geq 0$ ϵ-Übergängen

anwenden.

4. Die gesuchte Menge $Akt_A(aba) = \{5, 2, 1, 3\}$ ergibt sich daraus, daß wir auf die Zustände $2, 3, 4$ aus $Akt_A(ab)$ (wo möglich)

   - einen a-Übergang,
   - gefolgt von $k \geq 0$ ϵ-Übergängen

   anwenden.

5. Da $Akt_A(aba)$ den Finalzustand 5 enthält, wird aba akzeptiert.

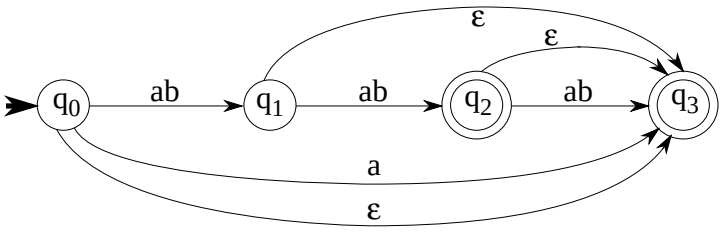
Bemerkung 2.24 Im Hinblick auf die Zeitkomplexität des Erkennungsverfahrens ist die zweite Methode zur Akzeptanzüberprüfung oft vorzuziehen. Betrachten wir als vereinfachtes Beispiel einen NDEA vom Typ 3 mit m Zuständen. Es sei k die maximale Zahl von Übergängen, die von einem Zustand $q \in Q$ wegführen. Im allgemeinen steigt die Zahl der beim Backtrackingverfahren zu betrachtenden Pfade exponentiell mit der Länge n des Eingabeworts an, sie kann k^n betragen. Beim zweiten Verfahren hingegen ergeben sich $n+1$ Schritte, wo wir die jeweilige Menge der aktiven Zustände aus der entsprechenden Vorgängermenge berechnen. Bei jedem Einzelschritt müssen wir für maximal m aktive Zustände der Vorgängermenge die möglichen Nachfolgerzustände für das betreffende Eingabesymbol berechnen und die erhaltenen Mengen zusammenfassen. Jede Einzelmenge hat maximal $k \leq m$ Elemente. Die Vereinigung kann in Zeit $O(km)$ berechnet werden, wenn die Menge sortiert vorliegen. Hier ergibt sich eine deutlich bessere Komplexität $O(nkm)$. Methoden, um den Rechenaufwand bei der Berechnung von Mengen aktiver Zustände bei NDEA vom Typ 2 klein zu halten, sind Gegenstand von Aufgabe 2.4.

2.4 Äquivalenz nicht-deterministischer und deterministischer endlicher Automaten - die Potenzmengenkonstruktion

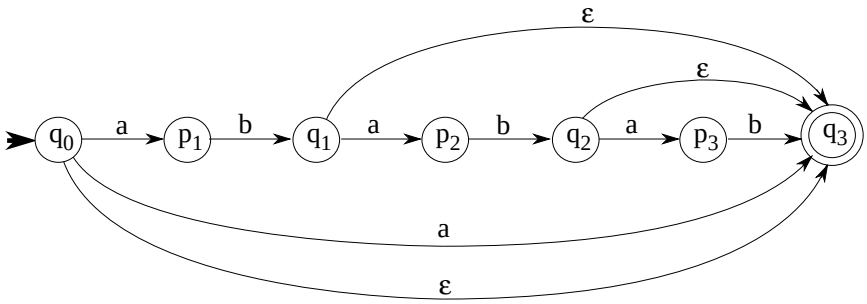
In diesem Abschnitt beschreiben wir eine Konstruktion, mit der man einen beliebigen NDEA A in einen DEA A' überführen kann, der dieselbe Sprache akzeptiert. Aus Gründen, die offenkundig werden, spricht man häufig von der *Potenzmengenkonstruktion*.

Definition 2.25 Zwei endliche Automaten A und A' heißen *äquivalent* genau dann, wenn $L(A) = L(A')$ gilt.

Die wesentlichen Ideen des Verfahrens seien zunächst an einem Beispiel erläutert. Als Ausgangspunkt wählen wir den folgenden NDEA A.



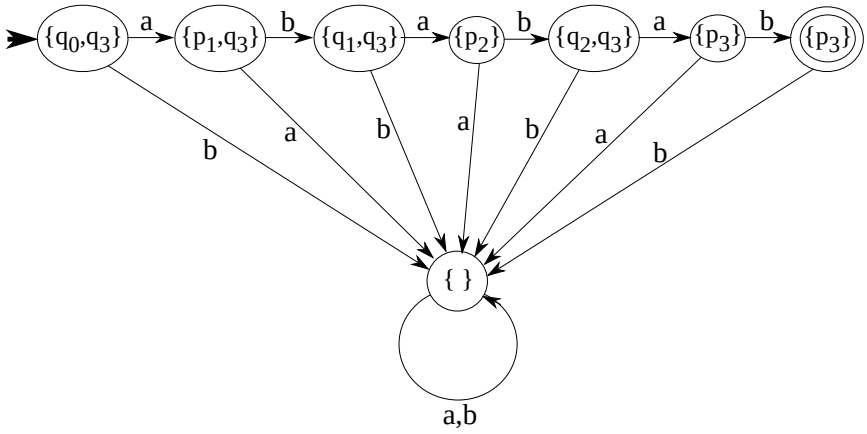
In einem ersten Schritt stellen wir einen äquivalenten NDEA A' her, wo alle Übergangslabels die Form $\sigma \in \Sigma$ oder ϵ haben. Dies kann einfach dadurch erreicht werden, daß all diejenigen Übergänge von A, die mit einem Wort der Länge $k \geq 2$ gelabelt sind, mittels neuer Zwischenzustände in k Übergänge unterteilt werden. In unserem Fall benötigen wir drei solcher Zwischenzustände, die wir mit p_1, p_2 und p_3 benennen. Der NDEA A' hat die folgende Form.



In einem zweiten Schritt wird nun intendierte deterministische Automat A'' konstruiert. Die wesentliche Idee ist es, daß

- Jeder Zustand von A'' für eine *Menge* möglicher Zustände von A' steht,
- Übergänge von A'' von einem Mengenzustand M immer *alle* von A'-Zuständen $q \in M$ *möglichen* A'-Übergänge zusammenfassen.

Dies soll näher anhand des nachfolgend dargestellten Ergebnisautomaten A'' erläutert werden.



Bevor wir überhaupt ein Symbol konsumiert haben, können wir in A' genau die Zustände q_0 und q_3 erreicht haben. Daher beginnen wir in A'' mit der Menge $\{q_0, q_3\}$, die beide Möglichkeiten zusammenfaßt und den Startzustand darstellt. Nach dem Konsum des Symbols a können wir in A' entweder bei p_1 oder bei q_3 stehen. In A'' reichten wir daher einen a-Übergang vom Start zum neuen Zustand $\{p_1, q_3\}$ ein. Tritt hingegen das Symbol b als erstes auf, so können wir in A' weder von q_0 noch von q_3 einen Übergang machen. In A' simulieren wir dies, indem wir mit b von $\{q_0, q_3\}$ in den Fallenzustand $\{\ \}$ gehen. Die Konstruktion wird in derselben Weise fortgesetzt. Für jeden Zustand M von A'' führen wir einen a-Übergang zur Menge

$$\{q \in Q' \mid \exists p \in M\text{: } q \text{ ist in } A' \text{ von } p \text{ durch Konsum von } a \text{ erreichbar}\}$$

ein, für die b-Übergänge verfahren wir mutatis mutandis.

Um die Idee zu formalisieren, sind die folgenden Begriffe nützlich.

Definition 2.26 [NF, SNF] Sei $A = (Q, \Sigma, \Delta, s, F)$ ein NDEA; für $w \in \Sigma^*$, $p \in Q$ und $P \subseteq Q$ definieren wir

- $SNF_A(p) = \{q \in Q : \ (p, \epsilon) \Longrightarrow_A^* (q, \epsilon)\}$ als die Menge der *Spontannachfolger* des Zustands p,

- $SNF_A(P) = \bigcup_{p \in P} SNF_A(p)$ als die Menge der *Spontannachfolger* der Zustandsmenge P,
- $NF_A(p, w) = \{q \in Q : \ (p, w) \Longrightarrow_A^* (q, \epsilon)\}$ als die Menge der *w-Nachfolger* des Zustands p,
- $NF_A(P, w) = \bigcup_{p \in P} NF_A(p, w)$ als die Menge der *w-Nachfolger* von Zuständen aus P.

Wir nennen eine Menge $P \subseteq Q$ *abgeschlossen unter Spontannachfolgern* genau dann, falls für alle $p \in P$ gilt: $SNF_A(p) \subseteq P$.

Es folgt leicht

Lemma 2.27 *1. $P \subseteq Q$ ist unter Spontannachfolgern abgeschlossen genau dann, wenn $SNF_A(P) = P$,*

2. Die Mengen $SNF_A(p)$, $SNF_A(P)$, $NF_A(p, w)$ und $NF_A(P, w)$ sind stets unter Spontannachfolgern abgeschlossen.

3. Falls für alle $(p, w, q) \in \Delta$ stets $|w| \leq 1$ gilt: Für alle $w \in \Sigma^*$ und $\sigma \in \Sigma$ gilt $NF_A(NF_A(P, w), \sigma) = NF_A(P, w\sigma)$.

Nach dieser Vorbereitung können wir zeigen:

Theorem 2.28 *Zu jedem NDEA $A = (Q, \Sigma, \Delta, s, F)$ kann man algorithmisch einen äquivalenten DEA $A' = (Q', \Sigma', \delta, s', F')$ konstruieren.*

Beweis: Sei $A = (Q, \Sigma, \Delta, s, F)$ gegeben. Wir gehen einfachheitshalber davon aus, daß jeder Übergang von A mit einem Wort der Länge $k \leq 1$ gelabelt ist (A hat Typ 2). Wie im obigen Beispiel gezeigt, ist diese Form im Bedarfsfall durch eine einfache Hilfskonstruktion zu erreichen. Wir definieren $A' = \{Q', \Sigma, \delta, s', F'\}$ wie folgt

- $Q' := \{P \subseteq Q \mid P$ abgeschlossen unter Spontannachfolgern bzgl. $A\}$,
- $s' := SNF_A(s)$,
- $F' := \{P \in Q' \mid F \cap P \neq \emptyset\}$ und

- $\delta(P, \sigma) := NF_A(P, \sigma)$ für alle $P \in Q'$.

Das obige Lemma zeigt, daß das Bild $\delta(P, \sigma)$ stets eine Menge in Q' ist. Dies zeigt, daß A' in der Tat ein deterministischer endlicher Automat ist. Es bleibt noch die Äquivalenz $L(A') = L(A)$ nachzuweisen. Dazu zeigen wir folgende

Hilfsbehauptung: Für alle $P \in Q', w \in \Sigma^*$ gilt $\hat{\delta}(P, w) = NF_A(P, w)$.

Beweis: Induktion über die Länge von w:

Induktionsbeginn: Es gelte $|w| = 0$, also $w = \epsilon$:
Es gilt $\hat{\delta}(P, \epsilon) = P$ da A' DEA ist. Andererseits folgt mit Lemma 2.27 Teil 1 auch $NF_A(P, \epsilon) = SNF_A(P) = P$.

Induktionsannahme: Die Behauptung sei für alle Wörter w der Länge $\leq n$ bereits bewiesen.

Induktionsschluß: Sei $w = \sigma w'$, $\sigma \in \Sigma$ und $\mid w' \mid = n$: Durch Anwendung der Definition der verallgemeinerten Übergangsfunktion $\hat{\delta}$, der Induktionshypothese, Einsetzen der Definition von δ und mittels Lemma 2.27 erhalten wir

$$\begin{aligned} \hat{\delta}(P, w'\sigma) &= \delta(\hat{\delta}(P, w'), \sigma) \\ &= NF_A(\delta(P, w'), \sigma) \\ &= NF_A(NF_A(P, w'), \sigma) \\ &= NF_A(P, w'\sigma), \end{aligned}$$

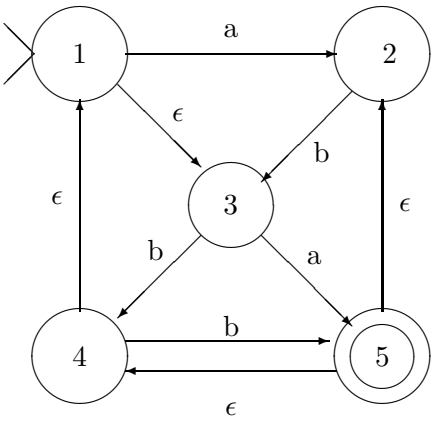
was den Beweis der Hilfsbehauptung beendet. Damit folgt nun

$$\begin{aligned} w \in L(A) &\iff \exists f \in F : f \in NF_A(s, w) \\ &\iff \exists f \in F : f \in NF_A(SNF(s), w) \\ &\iff F \cap NF_A(SNF(s), w) \neq \emptyset \\ &\iff NF_A(SNF_A(s), w) \in F' \\ &\iff NF_A(s', w) \in F' \\ &\iff \hat{\delta}(s', w) \in F' \\ &\iff w \in L(A'). \end{aligned}$$

Dies beendet den Beweis. □

Bemerkung 2.29 Man könnte für Q' auch $\mathcal{P}(\mathcal{Q})$ setzen. Aber: Mengen $P \subseteq Q$, die nicht unter SNF abgeschlossen sind, wären in A' von s' aus unerreichbar. Bei der **praktischen Konstruktion** von A' kann man sich von vornherein auf Zustände beschränken, die von s' aus erreichbar sind. Man wird also zunächst s' bestimmen. Für die Übergangsfunktion muß man dann die einzelnen σ-Übergänge zu s' bestimmen. Dabei erhält man neue (erreichbare) Zustände, die man dann in derselben Weise bearbeitet. Durch Iteration erhält man alle nötigen Zustände von A' und die Übergangsfunktion. Dies läuft genau auf eine komplette Vorberechnung aller für eine beliebige Eingabe möglichen Mengen aktiver Zustände des nichtdeterministischen Eingabeautomaten hinaus!

Beispiel:



SNF(1)= $\{1, 3\}$

SNF(2)= $\{2\}$

SNF(3)= $\{3\}$

SNF(4)= $\{1, 3, 4\}$

SNF(5)= $\{1, 2, 3, 4, 5\}$

Abbildung 2.4: Automat

- $s'' = \{1, 3\}$
- von 1,3 durch a erreichbar: 2,5. SNF(2,5)=$\{1, 2, 3, 4, 5\}$
- von 1,3 durch b erreichbar: 4. SNF(4)=$\{1, 3, 4\}$
- von 1,3,4 durch a erreichbar: 2,5. SNF(2,5)=$\{1, 2, 3, 4, 5\}$

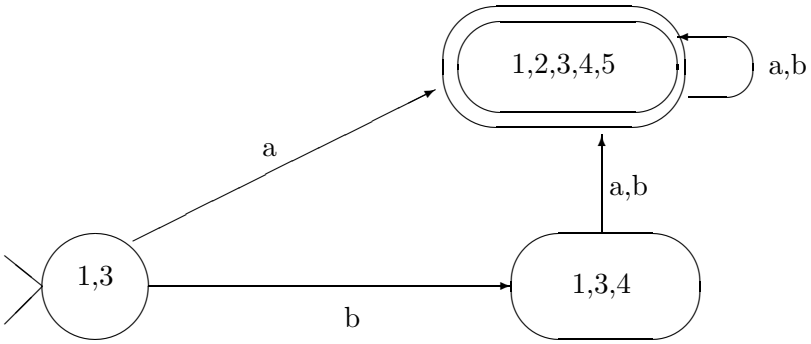


Abbildung 2.5: Automat

- ...

Definition 2.30 Sei $A = (Q, \Sigma, \delta, s, F)$ ein DEA. Ein Zustand $q \in Q$ heißt *erreichbar* genau dann, wenn ein Eingabewort $w \in \Sigma^*$ existiert, wo $\hat{\delta}(s, w) = q$. Ein Zustand $q \in Q$ heißt *terminalisierbar* genau dann, wenn ein Eingabewort $w \in \Sigma^*$ existiert, wo $\hat{\delta}(q, w)$ ein Finalzustand ist.

Ist ein nichtdeterministischer endlicher Automat mit n Zuständen gegeben, so kann der durch die Potenzmengenkonstruktion erreichte äquivalente deterministische Automat 2^n Zustände haben. Aus diesem Grund scheitert die praktische Ausführung bereits bei nichtdeterministischen Automaten einer relativ geringen Größe. In manchen Fällen läßt sich eine Milderung des Speicherplatzproblems dadurch erreichen, daß man sich bei der Konstruktion des deterministischen Automaten von vorneherein auf solche Zustände beschränkt, die vom Startzustand s' aus erreichbar sind. Im allgemeinen läßt sich aber die exponentielle Zustandsexplosion aus prinzipiellen Gründen nicht vermeiden.

Bemerkung 2.31 [vgl. [**?**]] Sei L_n die Menge aller Wörter der Länge $\geq n$ über dem Alphabet $\Sigma = \{0, 1\}$, wo der n-letzte Buchstabe eine 1 ist. Dann kann L_n mit einem NDEA mit $n+1$ Zuständen erkannt werden. Es läßt sich aber zeigen, daß jeder DEA, der L_n akzeptiert, mindestens 2^n Zustände hat.

Hingegen kann man zu einem NDEA A einen äquivalenten NDEA A' mit derselben Zustandszahl wie A anzugeben, der keine ϵ-Übergänge hat.

Lemma 2.32 *Zu jedem NDEA $A = (Q, \Sigma, \Delta, s, F)$ gibt es einen äquivalenten NDEA A', der dieselbe Zustandsmenge und denselben Startzustand hat, aber keine ϵ-Übergänge.*

Beweis. Ist $q \in Q$, so nennen wir einen Zustand $q_0 \in Q$ einen ϵ-Vorgänger von q, falls q ausgehend von q_0 in A durch eine (möglicherweise leere) Folge von ϵ-Übergängen erreichbar ist. Wir fügen nun ausgehend von A und für jeden Übergang (q, w, q') von Δ mit $|w| \geq 1$ für jeden ϵ-Vorgänger q_0 von q einen Übergang (q_0, w, q') zu Δ hinzu. Abschließend eliminieren wir alle Übergänge (q, ϵ, q') von Δ. Dies ergibt die Übergangsrelation Δ' von A'. Eine Zustand $q \in Q$ wird ein Finalzustand von A' genau dann, wenn q ein ϵ-Vorgänger eines Finalzustands von A ist.

Ist nun $w \in L(A)$, so gibt es in A eine Folge von Konfigurationen

$$\begin{aligned}
(s, w) &\Longrightarrow_A^* (p_1, w) \\
&\Longrightarrow_A^1 (q_1, w_1) \\
&\cdots \\
&\Longrightarrow_A^* (p_k, w_{k-1}) \\
&\Longrightarrow_A^1 (q_k, \epsilon) \\
&\Longrightarrow_A^* (f, \epsilon)
\end{aligned}$$

mit $f \in F$, wo jeweils w_{i+1} ein echtes Suffix von w_i ist und $w_{k-1} \neq \epsilon$. In A' erhalten wir

$$\begin{aligned}
(s, w) &\Longrightarrow_{A'}^1 (q_1, w_1) \\
&\cdots \\
&\Longrightarrow_{A'}^1 (q_k, \epsilon)
\end{aligned}$$

Da q_k Finalzustand von A', folgt $w \in L(A')$. Demnach gilt $L(A) \subseteq L(A')$. Ähnlich sieht man, daß auch umgekehrt $L(A') \subseteq L(A)$ gilt. ■

2.5 Von regulären Ausdrücken zu endlichen Automaten

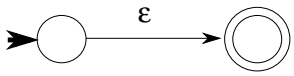
In diesem Abschnitt beschreiben wir zwei Methoden, um zu einem gegebenen regulären Ausdruck einen endlichen Automaten zu berechnen, der dessen Sprache erkennt. Hieraus ergibt sich inbesondere, daß jede reguläre Sprache durch einen endlichen Automaten erkennbar ist. Die Umkehrung, die besagt, dass die Sprache jedes endlichen Automaten regulär ist, wird in einem späteren Kapitel bewiesen.

2.5.1 Die Thompson-Konstruktion

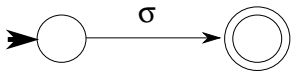
Um zu einem gegebenen regulärer Ausdruck α über dem Alphabet Σ einen NDEA A mit $L(A) = L(\alpha)$ zu berechnen, greifen wir auf folgende Teilkonstruktionen zurück. Es ergibt sich in jedem Schritt immer ein Automat mit genau einem Finalzustand, der vom Startzustand verschieden ist.

1. Ein möglicher NDEA A für den regulären Ausdruck 0 hat die Form

2. Ein möglicher NDEA A für den regulären Ausdruck 1 hat die Form

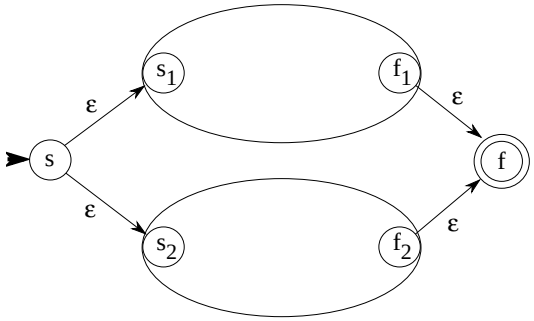


3. Ein möglicher NDEA A für den regulären Ausdruck σ ($\sigma \in \Sigma$) hat die Form

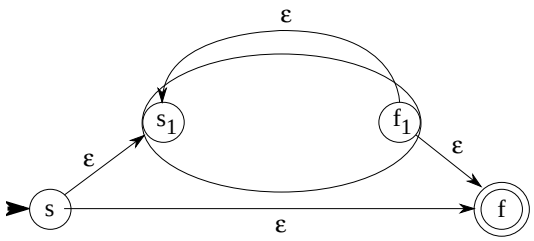


4. (Konkatenation) Sind $A_1 = (Q_1, \Sigma, s_1, \{f_1\}, \Delta_1)$ und $A_2 = (Q_2, \Sigma, s_2, \{f_2\}, \Delta_2)$ NDEA für die regulären Ausdrücke α_1 und α_2 mit disjunkten Zustandsmengen, so ist $A := (Q_1 \cup Q_2, \Sigma, s_1, \{f_2\}, \Delta_1 \cup \Delta_2') \cup \{(f_1, \epsilon, s_2)\}$ ein NDEA für $(\alpha_1 \cdot \alpha_2)$.

5. (Vereinigung) Sind A_1 und A_2 NDEA für die regulären Ausdrücke α_1 und α_2 wie unter 3., so ist $A := (Q_1 \cup Q_2 \cup \{s, f\}, \Sigma, s, \{f\}, \Delta_1 \cup \Delta_2 \cup \{(s, \epsilon, s_1), (s, \epsilon, s_2), (f_1, \epsilon, f), (f_2, \epsilon, f)\})$ ein NDEA für $(\alpha_1 + \alpha_2)$. Hierbei sind s und f neue Zustände.



6. (Kleene-Stern Ist $A_1 = (Q_1, \Sigma, s_1, \{f_1\}, \Delta_1)$ ein NDEA für den regulären Ausdruck α_1, so ist $A := (Q_1 \cup \{s, f\}, \Sigma, s, \{f\}, \Delta_1 \cup \{(s, \epsilon, s_1), (s, \epsilon, f), (f_1, \epsilon, s_1), (f_1, \epsilon, f)\})$ ein NDEA für α_1^*. Hierbei sind s und f neue Zustände.



Als eine offenkundige Folgerung erhalten wir das folgende Korollar.

Korollar 2.33 *Jede reguläre Sprache ist durch einen endlichen Automaten erkennbar. Es gibt einen Algorithmus, um zu einem gegebenen regulären Ausdruck α einen EA A mit $L(A) = L(\alpha)$ zu konstruieren.*

Bemerkung 2.34 Wenn wir mit der oben dargestellten Methode von Thompson einen NDEA A für den regulären Ausdruck α konstruieren, so hat A die nachfolgenden Eigenschaften. Hierbei bezeichne m die Zahl der Symbole der Form $\sigma, 0, 1, \cup, \cdots, *$ von α.

1. Die Gesamtzahl der Zustände von A ist kleinergleich $2m$.
2. Die Gesamtzahl der Übergänge von A ist kleinergleich $4m$.
3. A kann aus α in Zeit $O(m)$ berechnet werden.
4. Es gibt eine Nummerierung der Zustände von A in der Form $Q = \{1, 2, \ldots, n\}$ so daß jeder nichtleere Übergang die Form $(i, \sigma, i+1)$ für ein $i \in \{1, \ldots, n-1\}$ hat.

2.5.2 Die Berry-Sethi Konstruktion

Die im vorigen Abschnitt dargestellte Konstruktion von Thompson hat Vor- und Nachteile. Positiv ist die lineare Berechnungszeit des Automaten zu einem gegebenen regulären Ausdruck. Im Rahmen praktischen Anwendungen stellt die Vielzahl von leeren Übergängen, die entstehen können, einen signifikanten Nachteil dar. Sie erschwert auch eine anschließende Determinisierung mit Hilfe der Potenzmengenkonstruktion. Wir gehen nachfolgend auf eine zweite Konstruktions ein, bei der leere Übergänge vermieden werden. Die Konstruktion geht auf Glushkov zurück und wurde durch Berry und Sethi bekannt gemacht.

Sei α ein regulärer Ausdruck, zu dem ein endlicher Automat A mit $L(A) = L(\alpha)$ konstruiert werden soll. Wir nummerieren zunächst alle Vorkommen von Symbolen aus Σ in α in der Reihenfolge ihres Auftretens in α von links nach rechts durch. Jedes Symbolvorkommen ist damit mit einer eindeutigen Nummer aus $\{1, \ldots, n\}$ versehen, wo n die Gesamtzahl der Vorkommen beschreibt.

Von α gelangen wir zum regulären Ausdruck $\bar{\alpha}$ indem wir jedes Vorkommen eines Symbols aus Σ explizit mit seiner Nummer als Index versehen. Der reguläre Ausdruck $\bar{\alpha}$ ist über dem Alphabet $\bar{\Sigma}$ erklärt, das aus allen in dieser Weise explizit indexierten Symbolen besteht. Die Zahlen $\{1, \ldots, n\}$ können als Positionen im Ausdruck $\bar{\alpha}$ betrachtet werden.

Beispiel 2.35 Es sei α der Ausdruck $(a+b)^*(ab)^*(ac)^*$. Dann hat $\bar{\alpha}$ die Form $(a_1 + b_2)^*(a_3 b_4)^*(a_5 c_6)^*$.

Wir konstruieren nun zunächst einen DEA $\bar{A}$ mit $L(\bar{A}) = L(\bar{\alpha})$. Später ergibt sich der gesuchte NDEA A einfach dadurch, daß wir bei den Über-

gangslabels von $\bar{A}$ alle Symbolindices aus $\{1,\ldots,n\}$ weglöschen. Zur Definition von $\bar{A}$ werden die folgenden Hilfsmengen benötigt. Nachfolgend bezeichne τ_i das eindeutig bestimmte Symbol aus $\bar{\Sigma}$ mit Index i $(1 \leq i \leq n)$.

$$\begin{aligned} \mathit{First}(\bar{\alpha}) &:= \{i \in \{1,\ldots,n\} \mid \exists w \in \bar{\Sigma}^* : \tau_i w \in L(\bar{\alpha})\} \\ \mathit{Last}(\bar{\alpha}) &:= \{i \in \{1,\ldots,n\} \mid \exists w \in \bar{\Sigma}^* : w\tau_i \in L(\bar{\alpha})\} \\ \mathit{Follow}(\bar{\alpha}, i) &:= \{j \in \{1,\ldots,n\} \mid \exists w, v \in \bar{\Sigma}^* : w\tau_i\tau_j v \in L(\bar{\alpha})\} \end{aligned}$$

Die Menge $\mathit{First}(\bar{\alpha})$ (resp. $\mathit{Last}(\bar{\alpha})$) gibt diejenigen Positionen an, die zum ersten (letzten) Symbol eines Worts aus $L(\bar{\alpha})$ korrespondieren können. Die Menge $\mathit{Follow}(\bar{\alpha}, i)$ gibt diejenigen Positionen an, die auf die i-te Position beim Lesen folgen können.

Beispiel 2.36 Es habe $\bar{\alpha}$ die Form $(a_1 + b_2)^*(a_3b_4)^*(a_5c_6)^*$. Dann ist $\mathit{First}(\bar{\alpha}) = \{1,2,3,5\}$ und $\mathit{Last}(\bar{\alpha}) = \{1,2,4,6\}$. Weiter gilt

$$\begin{aligned} \mathit{Follow}(\bar{\alpha}, 1) &= \{1,2,3,5\}, \\ \mathit{Follow}(\bar{\alpha}, 2) &= \{1,2,3,5\}, \\ \mathit{Follow}(\bar{\alpha}, 3) &= \{4\}, \\ \mathit{Follow}(\bar{\alpha}, 4) &= \{3,5\}, \\ \mathit{Follow}(\bar{\alpha}, 5) &= \{6\}, \\ \mathit{Follow}(\bar{\alpha}, 6) &= \{5\}. \end{aligned}$$

Es bezeichne $\mathit{Empty}(\alpha)$ die Menge $L(\alpha) \cap \{\epsilon\}$. Rekursiv kann diese Menge wie folgt berechnet werden:

$$\begin{aligned} \mathit{Empty}(0) &= \emptyset, \\ \mathit{Empty}(1) &= \{\epsilon\}, \\ \mathit{Empty}(\sigma) &= \emptyset (\sigma \in \Sigma), \\ \mathit{Empty}(\alpha + \beta) &= \mathit{Empty}(\alpha) \cup \mathit{Empty}(\beta), \\ \mathit{Empty}(\alpha\beta) &= \mathit{Empty}(\alpha) \cap \mathit{Empty}(\beta), \\ \mathit{Empty}(\alpha^*) &= \{\epsilon\}. \end{aligned}$$

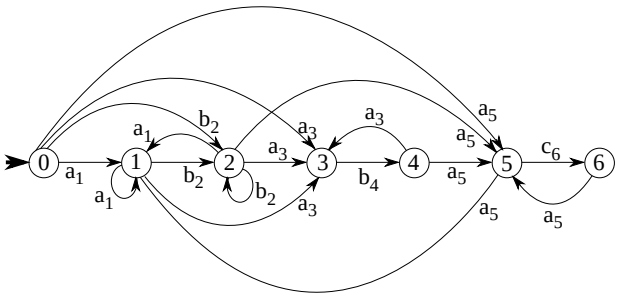


Abbildung 2.6: Automat $\bar{A}$ zu Beispiel 2.37.

Nach dieser Vorbereitung können wir den DEA $\bar{A}$ definieren als $\bar{A} := (Q, \bar{\Sigma}, s, F, \bar{\delta})$ wo gilt

1. $Q := \{0, 1, \ldots, n\}$ enthält alle Positionen von $\bar{\alpha}$ sowie die 0,
2. $s := 0$ ist der Startzustand,
3. $F := \mathit{Last}(\bar{\alpha})$ falls $\mathit{Empty}(\alpha) = \emptyset$ und $F := \mathit{Last}(\bar{\alpha}) \cup \{0\}$ falls $\mathit{Empty}(\alpha) = \{\epsilon\}$,
4. $\bar{\delta}(i, \_)$ ist für alle Zustände $i \in \{1, \ldots, n\}$ und alle Symbole τ_j mit $j \in \mathit{Follow}(\bar{\alpha}, i)$ definiert durch $\bar{\delta}(i, \tau_j) := j$. Weiter ist auch $\bar{\delta}(0, \_)$ für alle Symbole τ_i mit $i \in \mathit{First}(\bar{\alpha})$ definiert durch $\bar{\delta}(0, \tau_i) := i$.

Beispiel 2.37 Wie oben habe $\bar{\alpha}$ die Form $(a_1 + b_2)^*(a_3 b_4)^*(a_5 c_6)^*$. Dann ist $\bar{A}$ der in Abbildung 2.6 dargestellte DEA.

Es ist leicht zu sehen, daß die aus der Berry-Sethi Konstruktion erhaltenen Automaten folgende Eigenschaft haben: für jeden Zustand q existiert ein eindeutig bestimmtes Symbol $\sigma \in \Sigma$, so daß jeder in q endende Übergang mit dem Label q versehen ist.

2.6 Weitere Konstruktionen, Abschlußeigenschaften und Entscheidungsfragen für endliche Automaten

2.6.1 Die Produktkonstruktion

Mit einem eleganten Verfahren kann man zu zwei gegebenen *deterministischen* EA A_1 und A_2 direkt einen DEA A konstruieren, so daß $L(A) = L(A_1) \cup L(A_2)$ (bzw. $L(A) = L(A_1) \cap L(A_2)$) gilt. Gilt $A_i = (Q_i, \Sigma, \delta_i, s_i, F_i)$ $(i = 1, 2)$, so definiert man A als den DEA mit

1. Zustandsmenge $Q_1 \times Q_2$,
2. Startzustand $s := (s_1, s_2)$,
3. Finalzustandsmenge $F := \{(q_1, q_2) \in Q_1 \times Q_2 \mid q_1 \in F_2 \text{ oder } q_2 \in F_2\}$,
4. Übergangsfunktion $\delta((q_1, q_2), a) := (\delta_1(q_1, a), \delta_2(q_2, a))$.

Dieser sogenannte „Produktautomat“ von A_1 und A_2 simuliert einen parallelen Erkennungsprozeß in A_1 und A_2. Akzeptiert werden diejenigen Wörter w, die in zumindest einem der beiden parallelen Abläufe in einen Finalzustand des betreffenden Komponentenautomaten führen. Dies zeigt, daß $L(A) = L(A_1) \cup L(A_2)$ gilt. Ändert man die Definition der Finalzustandsmenge ab zu

3b. $F := \{(q_1, q_2) \in Q_1 \times Q_2 \mid q_1 \in F_2 \text{ und } q_2 \in F_2\}$,

so werden genau die Wörter akzeptiert, die in beiden Komponentenautomaten akzeptiert werden, es folgt $L(A) = L(A_1) \cap L(A_2)$. Auch die Differenzen $L(A_1) \setminus L(A_2)$ und $L(A_2) \setminus L(A_1)$ können durch entsprechend modifizierte Produktautomaten erkannt werden. Für die erste Differenz definieren wir

3c. $F := \{(q_1, q_2) \in Q_1 \times Q_2 \mid q_1 \in F_2 \text{ und } q_2 \notin F_2\}$.

2.6.2 Abschlusseigenschaften der Klasse der von endlichen Automaten akzeptierten Sprachen

Satz 2.38 *Die Klasse der von EA akzeptierten Sprachen über dem Alphabet Σ ist abgeschlossen unter*

1. *Konkatenation,*
2. *Kleene-Stern,*
3. *Vereinigung,*
4. *Komplement*
5. *Mengendifferenz und*
6. *Durchschnitt.*

Beweis: Offenkundig ist die Sprache Σ^* von einem DEA A^Σ erkennbar. Es seien A_1 und A_2 zwei EA. Wir dürfen annehmen, daß beide DEA sind (Potenzmengenkonstruktion). Die Produktkonstruktion zeigt, dass die Sprachen $L(A_1) \cup L(A_2)$, $L(A_1) \cap L(A_2)$, $L(A_1) \setminus L(A_2)$, $\Sigma^* \setminus L(A_1)$ von endlichen Automaten erkannt werden. Eine offenkundige Verallgemeinerung der Thompsonkonstruktion zeigt, daß stets $L(A_1) \circ L(A_2)$ sowie $L(A_1)^*$ durch EA erkannt werden. ■

2.6.3 Entscheidbarkeitsfragen für endliche Automaten

Als eine einfache Folgerung der Konstruktionsprinzipien für endliche Automaten kann man zeigen, daß viele Fragen über endliche Automaten entscheidbar sind.

Satz 2.39 *Es gibt Algorithmen, die zu gegebenen Automaten A, A_1 und A_2 entscheiden:*

1. *gegeben $w \in \Sigma^*$, gilt $w \in L(A)$?*

2. *Gilt* $L(A) = \emptyset$*?*
3. *Gilt* $L(A) = \Sigma^*$*?*
4. *Gilt* $L(A_1) \subseteq L(A_2)$*?*
5. *Gilt* $L(A_1) = L(A_2)$*?*

Beweis:

1. trivial.
2. Kennzeichne alle von s aus erreichbaren Zustände. Ist ein Finalzustand gekennzeichnet, so ist $L(A) \neq \emptyset$, sonst nicht.
3. OBdA ist A ein DEA. Konstruiere zunächst EA A' für $\Sigma^* \setminus L(A)$. Dann gilt $L(A) = \Sigma^* \iff L(A') = \emptyset$.
4. $L(A_1) \subseteq L(A_2) \iff (\Sigma^* \setminus L(A_2)) \cap L(A_1) = \emptyset$.
5. $L(A_1) = L(A_2) \iff L(A_1) \subseteq L(A_2) \bigwedge L(A_2) \subseteq L(A_1)$.

2.7 Kleenes Theorem

Wir können nun den Zusammenhang zwischen regulären Sprachen und endlichen Automaten abschließend darstellen.

Theorem 2.40 (Kleenes Theorem) *Die Klasse der regulären Sprachen ist genau die Klasse von Sprachen, die durch einen EA akzeptiert werden.*

Beweis: Sei α ein regulärer Ausdruck. Wie wir bereits wissen, existiert ein endlicher Automat A mit $L(A) = L(\alpha)$.

Sei nun A ein endlicher Automat. Wir können OBdA annehmen, daß $A = (Q, \Sigma, \delta, s, F)$ ein DEA ist. Wir numerieren nun die Symbole des Eingabealphabets $\Sigma = \{\sigma_1, \cdots, \sigma_n\}$ und der Zustandsmenge $Q = \{q_1, \cdots, q_k\}$. Hierbei gelte $s = q_1$. Für $q \in Q$ setze $L_A(q) := \{w \in \Sigma^* : \ \hat{\delta}(q, w) \in F\}$. Es ist $L_A(q)$ die Menge der vom „neuen Startzustand“ q aus in A akzeptierten Worte.

Wir führen nun x_i als Variable für einen noch zu bestimmenden regulären Ausdruck, der $L_A(q_i)$ repräsentiert ($1 \leq i \leq k$). Ist $\delta(q_i, \sigma_j) = q_h$, so bezeichne $x_{i,j}$ die Variable x_h. Ferner sei $y_i := \emptyset^*$ falls $q_i \in F$, $y_i := \emptyset$ sonst. Nun betrachten wir das System von Gleichungen

$$x_i \equiv \bigcup_{j=1,\ldots,n} \sigma_j x_{i,j} \cup y_i \quad (\text{ für } i = 1, \cdots, k).$$

Wenn man die Form der Übergangsfunktion des Automaten beachtet, so sieht man, daß die Gleichungen offenkundig gültige Äquivalenzen zwischen den gesuchte regulären Ausdrücken für die Sprachen $L_A(q_i)$ beschreiben ($i = 1, \ldots, k$). Das nachfolgende Vorgehen entspricht dem Auflösen linearer Gleichungssysteme mit k Unbestimmten über den reellen Zahlen. Falls x_1 in der rechten Seite von $x_1 \equiv \bigcup_{j=1,\cdots,n} \sigma_j x_{1,j} \cup y_1$ noch auftritt, etwa beim Koeffizienten σ_a, so formen wir unter Verwendung von Lemma 2.7 die Gleichung zu

$$x_1 \equiv \sigma_a x_{1,a} \cup \bigcup_{j=1,\cdots,n; j \neq a} \sigma_j x_{1,j} \cup y_1$$

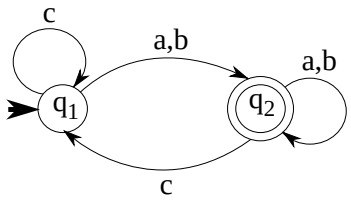
um. Hieraus ergibt sich nach Lemma 2.7, Teil 8 nun

$$x_1 \equiv (\sigma_a)^* (\bigcup_{j=1,\cdots,n; j \neq a} \sigma_j x_{1,j} \cup y_1)$$

Man beachte, daß die in Lemma 2.7, Teil 8 geforderte Voraussetzung erfüllt ist ($\epsilon \notin L(\sigma_a)$). Durch Wiederholen erhält man eine Gleichung, in der x_1 rechts eliminiert ist. Nun kann jedes weitere Vorkommen von x_1 in den rechten Seiten der restlichen Gleichungen durch Ersetzen eliminiert werden.

Durch gleiche Behandlung der restlichen Gleichungen können alle Variablen aus der rechten Seite der Gleichung für x_1 eliminiert werden. Es ist nun x_1 ist der gesuchte reguläre Ausdruck für $L(A)$. ■

Beispiel 2.41 Wir betrachten den folgenden deterministischen endlichen Automaten:



Das zugehörige Gleichungssystem hat die zwei Gleichungen

$$\begin{aligned} x_1 &\equiv ax_2 + bx_2 + cx_1 + \emptyset \\ x_2 &\equiv ax_2 + bx_2 + cx_1 + \emptyset^* \end{aligned}$$

Wir entfernen nun zunächst x_1 aus den rechten Seiten.

$$\begin{aligned} x_1 &\equiv cx_1 + ax_2 + bx_2 + \emptyset = (c)^*(ax_2 + bx_2 + \emptyset) \\ x_2 &\equiv (a+b)x_2 + c(c)^*(ax_2 + bx_2 + \emptyset) + \emptyset^* \\ &\equiv (a + b + cc^*a + cc^*b)x_2 + \emptyset^* \end{aligned}$$

Hieraus ergibt sich

$$x_2 \equiv (a + b + cc^*a + cc^*b)^*\emptyset^*$$

Wir können nun x_2 analog aus den rechten Seiten entfernen.

$$\begin{aligned} x_1 &\equiv c^*(a+b)(a + b + cc^*a + cc^*b)^*\emptyset^* \\ &\equiv c^*(a+b)(a + b + cc^*a + cc^*b)^* \end{aligned}$$

Aus Satz 2.38 erhalten wir nun sofort die entsprechende Aussage über reguläre Sprachen.

Satz 2.42 *Die Klasse der regulären Sprachen über dem Alphabet Σ ist abgeschlossen unter*

1. *Konkatenation,*
2. *Kleene-Stern,*
3. *Vereinigung,*
4. *Komplement*
5. *Mengendifferenz und*
6. *Durchschnitt.*

2.8 Reduzierte endliche Automaten

Definition 2.43 [ununterscheidbar] Sei $A = (Q, \Sigma, \delta, s, F)$ ein DEA. Zwei Zustände $p, q \in Q$ heißen *k-ununterscheidbar* ($k \geq 0$) genau dann, wenn für alle $w \in \Sigma^*$ mit $|w| \leq k$ gilt: $\hat{\delta}(p, w) \in F \iff \hat{\delta}(q, w) \in F$. Zwei Zustände heißen *ununterscheidbar* genau dann, wenn sie für jedes $k \in N$ k-ununterscheidbar sind.

Beispiel 2.44 Bei dem in Abbildung 2.7 dargestellten Automaten A sind die Zustände s, p, q und x paarweise 0-ununterscheidbar; s und x sind

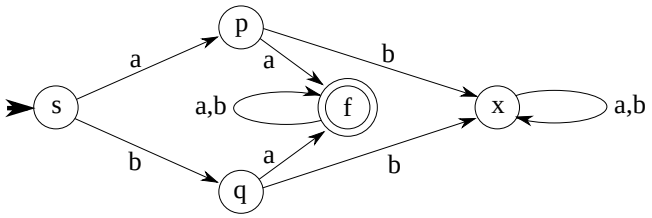
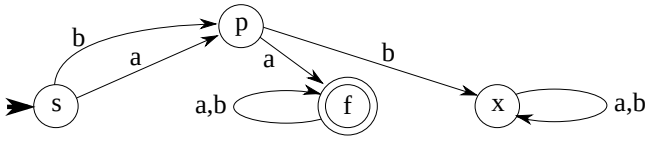


Abbildung 2.7: Automat mit ununterscheidbaren Zuständen p, q.

1-ununterscheidbar, aber nicht 2-ununterscheidbar; p und q sind ununterscheidbar.

Enthält ein Automat A zwei ununterscheidbare Zustände $p \neq q$, so kann stets einer der beiden eliminiert werden. Ist etwa q vom Startzustand verschieden, so löschen wir q. Alle Übergänge von Zuständen $r \neq q$ aus A, die in A auf q zeigen, weisen beim neuen Automaten auf p. Alle von q in A ausgehenden Übergänge werden gelöscht. Bei dem in Abbildung 2.7 dargestellten Automaten ergibt sich nach diesem Prinzip der folgende Automat:



Dieser Automat hat keine ununterscheidbaren Zustände mehr. Im allgemeinen bleiben weitere ununterscheidbaren Zustände. Durch Iteration erreicht man stets einen DEA, der keine ununterscheidbaren Zustände mehr hat.

Definition 2.45 [reduzierter DEA] Ein DEA A heißt *reduziert*, wenn je zwei verschiedene Zustände von A unterscheidbar sind und wenn jeder Zustand vom Startzustand aus erreichbar ist.

Wir wollen nun eine Konstruktion angeben, bei der in einem Schritt aus einem DEA $A = (Q, \Sigma, \delta, s, F)$ ein äquivalenter reduzierter DEA A' erreicht wird. Als Hilfsmittel charakterisieren wir zunächst k-Ununterscheidbarkeit sowie Ununterscheidbarkeit mit Hilfe der folgenden Äquivalenzrelationen. Für $p, q \in Q$ definieren wir induktiv

$$\begin{aligned} p \sim_0 q &\iff p, q \in F \quad \text{oder } p, q \notin F \\ p \sim_{i+1} q &\iff p \sim_i q \quad \text{und } \delta(p, \sigma) \sim_i \delta(q, \sigma) \, \forall \sigma \in \Sigma \end{aligned}$$

Lemma 2.46 *Sei $A = (Q, \Sigma, \delta, s, F)$ ein DEA. Für $i \geq 0$ sei die Äquivalenzrelation $\sim_i$ wie oben definiert.*

1. *Zwei Zustände p, q sind k-ununterscheidbar genau dann, wenn $p \sim_k q$ gilt.*

2. *Es gibt ein $n \leq |Q|$ so dass die Äquivalenzrelationen $\sim_{n+k}$ für alle $k \geq 0$ übereinstimmen.*

3. *Es sind p, q ununterscheidbar genau dann, wenn $p \sim_n q$.*

Beweis: Teil 1: Wir verwenden Induktion über k. Der Fall $k = 0$ ist trivial, da sich in Definition 2.43 aus $\mid w \mid \leq 0$ und $w = \epsilon$ die Bedingung $p \in F \iff q \in F$ ergibt.

Im Induktionsschritt nehmen wir an, dass die Äquivalenz aus Teil 1 für ein $k \geq 0$ und alle $p, q \in Q$ gilt. Sind nun p, q als Zustände $k+1$-ununterscheidbar, so folgt:

1. p, q sind k-ununterscheidbar (da ja alle Wörter der Länge $\leq k$ insbesondere Länge $\leq k+1$ haben),

2. für jedes $\sigma \in \Sigma$ sind $\delta(p, \sigma)$ und $\delta(q, \sigma)$ stets k-ununterscheidbar (da für jedes Wort w' der Länge $\leq k$ stets $\sigma w'$ Länge $\leq k+1$ hat).

Nach Induktionsvoraussetzung gilt damit $p \sim_k q$ und $\delta(p,\sigma) \sim_k \delta(q,\sigma)$ für jedes $\sigma \in \Sigma$, woraus $p \sim_{k+1} q$ folgt. Die Umkehrung zeigt man analog. Dies beendet den Induktionsbeweis von Teil 1.

Der Rest des Beweises bleibt als Übung offen. □

Lemma 2.47 *Zu jedem DEA A läßt sich algorithmisch ein äquivalenter und reduzierter DEA A' angeben. Enthält A zwei verschiedene ununterscheidbare Zustände, so hat A' echt weniger Zustände.*

Beweis: 1. Im ersten Schritt entfernen wir, sofern nötig, alle unerreichbaren Zustände. Wir nehmen nun an, daß in $A = (Q, \Sigma, \delta, s, F)$ alle Zustände $q \in Q$ vom Startzustand s erreichbar sind.

2. Wir berechnen nun die Folge der Äquivalenzrelationen $\sim_0, \sim_1, \ldots$ solange, bis wir die maximale Verfeinerung $\sim_n$ erreicht haben, für die dann $\sim_n = \sim_{n+1} = \ldots$ gilt (Lemma 2.46). Hierzu setzen wir zunächst $Q_{/\sim_0} = \{F, Q \setminus F\} = \{Q_1^{(0)}, Q_2^{(0)}\}$. Ist nun $Q_{/\sim_i} = \{Q_1^{(i)}, \cdots, Q_{l_i}^{(i)}\}$ bereits berechnet, so berechnen wir für alle $p \in Q_j^{(i)}$ die Äquivalenzklasse bezüglich $\sim_{i+1}$ in der Form $\{q \in Q_j^{(i)} \mid \forall \sigma \in \Sigma : \delta(p,\sigma) \sim_i \delta(q,\sigma)\}$. Hierdurch ergibt sich $Q_{/\sim_{i+1}}$.

3. Nachfolgend bezeichne $[q]$ die Äquivalenzklasse $\{p \in Q \mid \ p \sim_n q\}$. Wir definieren nun A' in der Form $A' := (Q', \Sigma, s', F', \delta')$ wo gilt

$$\begin{aligned} Q' &:= \{[q] \mid q \in Q\} \\ s' &:= [s] \\ F' &:= \{[f] \mid f \in F\} \\ \delta'([q],\sigma) &:= [\delta(q,\sigma)] \qquad (*) \end{aligned}$$

Zunächst muss nun verifiziert werden, dass δ' wohldefiniert ist: Es gelte $[q] = [q']$. Dann folgt $q \sim_n q'$ sowie nach Wahl von n auch $q \sim_{n+1} q'$, also $\delta(q,\sigma) \sim_n \delta(q',\sigma)$ und damit $\delta'([q],\sigma) = \delta'([q'],\sigma)$ für alle $\sigma \in \Sigma$.

Es folgt sofort, daß A' ein DEA ist. Wir zeigen als nächstes, daß A' reduziert ist. Sei $[q] \in Q'$. Nach Voraussetzung existiert ein Wort $w \in \Sigma^*$ mit $\hat{\delta}(s,w) = q$. Hieraus folgt gemäß $(*)$ auch $\hat{\delta}'([s],w) = [q]$, somit sind in A' alle Zustände vom Startzustand aus erreichbar.

Es seien nun $[p] \neq [q]$ zwei Zustände von A'. Dann sind p und q in A unterscheidbar (Wahl von n sowie Lemma 2.46). Es gibt somit OBdA ein

Wort $w \in \Sigma^*$ mit $\hat{\delta}(p,w) \in F$ und $\hat{\delta}(q,w) \notin F$. Offenkundig ist $\hat{\delta}(q,w)$ bezüglich $\sim_n$ zu keinem Finalzustand äquivalent, da bereits in $\sim_0$ Finalzustände und Nichtfinalzustände in verschiedenen Klassen liegen. Es folgt, daß $\hat{\delta}'([p],w)] = [\hat{\delta}(p,w)] \in F'$ und $\hat{\delta}'([q],w) = [\hat{\delta}(q,w)] \notin F'$. Damit sind $[p]$ und $[q]$ aber in A' unterscheidbar.

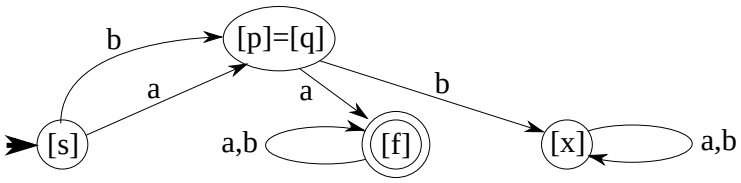
Wir zeigen die Äquivalenz von A und A'. Für alle $w \in \Sigma^*$ gilt

$$\begin{aligned} w \in L(A') &\Leftrightarrow \hat{\delta}'([s],w) \in F' \\ &\Leftrightarrow \exists f \in F : \hat{\delta}'([s],w) = [f] \\ &\Leftrightarrow \exists f \in F : \hat{\delta}(s,w) \text{ und } f \text{ sind ununterscheidbar} \\ &\Leftrightarrow \delta(s,w) \in F. \end{aligned}$$

Bei der dritten Äquivalenz wird (*) verwendet. Bei der letzten Äquivalenz nützen wir aus, dass jeder Zustand, der zu einem Finalzustand ununterscheidbar ist, selbst final sein muss.

Die verbleibende Aussage folgt unmittelbar daraus, dass die Zustände von A' Äquivalenzklassen ununterscheidbarer Zustände von A sind. □

Beispiel 2.48 Das Vorgehen lässt sich anhand des in Abbildung 2.7 dargestellten Automaten illustrieren. Die Äquivelenzklassen bezüglich $\sim_0$ sind $Q_1^{(0)} = \{s,p,q,x\}$ und $Q_2^{(0)} = \{f\}$.
Betrachte s: mit a nach $Q_1^{(0)}$, mit b nach $Q_1^{(0)}$.
Betrachte x: mit a nach $Q_1^{(0)}$, mit b nach $Q_1^{(0)}$.
Betrachte p: mit a nach $Q_2^{(0)}$, mit b nach $Q_1^{(0)}$.
Betrachte q: mit a nach $Q_2^{(0)}$, mit b nach $Q_1^{(0)}$.
Man erhält also bezüglich $\sim_1$ die Äquivalenzklassen $Q_1^{(1)} = \{s,x\}$, $Q_2^{(1)} = \{p,q\}$ und $Q_3^{(1)} = \{f\}$.
Betrachte s: mit a nach $Q_2^{(1)}$, mit b nach $Q_2^{(1)}$.
Betrachte x: mit a nach $Q_1^{(1)}$, mit b nach $Q_1^{(1)}$.
Betrachte p: mit a nach $Q_3^{(1)}$, mit b nach $Q_1^{(1)}$.
Betrachte q: mit a nach $Q_3^{(1)}$, mit b nach $Q_1^{(1)}$.
Man erhält also bezüglich $\sim_2$ die Äquivalenzklassen $Q_1^{(2)} = \{s\}$, $Q_2^{(2)} = \{x\}$, $Q_3^{(2)} = \{p,q\}$ und $Q_3^{(3)} = \{f\}$. Im nächsten Schritt ändern sich die Klassen nicht mehr. Als Ergebnis ergibt sich



Definition 2.49 Es sei L eine beliebige Sprache über dem Alphabet Σ. Auf der Menge aller Wörter Σ^* definieren wir die *Nerode-Äquivalenzrelation* $\sim_L$ bzgl. L in der Form

$$v_1 \sim_L v_1 \quad :\Leftrightarrow \quad (\forall w \in \Sigma^* : v_1 w \in L \text{ gdw. } v_2 w \in L).$$

Zwei Wörter sind v_1, v_2 sind im Nerode-Sinn äquivalent genau dann, wenn sie in genau denselben Rechtskontexten w "grammatikalisch" (d.h. in L) sind.

Theorem 2.50 (Satz von Nerode) *Es sei $L \subseteq \Sigma^*$ und $\sim_L$ wie oben. Dann ist L regulär gdw. die Zahl der $\sim_L$-Äquivalenzklassen von Σ^* endlich ist.*

Beweis. Es sei L regulär. Nach dem Theorem von Kleene existiert ein DEA $A = (Q, \Sigma, \delta, s, F)$ mit $L = L(A)$. Wir definieren für $q \in Q$ die Sprache $L_{A,s}(q) := \{w \in \Sigma^* \mid \hat{\delta}(s, w) = q\}$. Sind nun v_1, v_2 zwei Wörter aus $L_{A,s}(q)$ so gilt offenkundig $v_1 \sim_L v_1$. Daher kann die Zahl der $\sim_L$-Äquivalenzklassen nicht die Zahl der Zustände von A übersteigen.

Es sei nun umgekehrt $\{M_1, \ldots, M_n\}$ die endliche Menge aller $\sim_L$-Äquivalenzklassen. Die $\sim_L$-Äquivalenzklasse von $v \in \Sigma^*$ sei in der Form $[v]$ notiert. Wir definieren einen DEA A in der Form $A := (Q, \Sigma, \delta, s, F)$ mit $Q := \{M_1, \ldots, M_n\}$, $\delta([v], \sigma) := [v\sigma]$ für alle $v \in \Sigma^*, \sigma \in \Sigma$, $s := [\epsilon]$, $F = \{M_i \in Q \mid M_i \subseteq L\}$.

Offenkundig folgt aus $v_1 \sim_L v_2$ und $\sigma \in \Sigma$ stets auch $v_1\sigma \sim_L v_2\sigma$. Dies zeigt, daß δ wohldefiniert ist. Es ist damit A tatsächlich ein DEA. Weiter gilt für $w \in \Sigma^*$ nach Definition von δ stets

$$\begin{aligned} w \in L &\Leftrightarrow [w] \in F \\ &\Leftrightarrow \hat{\delta}([\epsilon], w) = \hat{\delta}(s, w) \in F \\ &\Leftrightarrow w \in L(A). \end{aligned}$$

Bei der ersten Äquivalenz beachte man, daß im Falle $w' \sim_L w$ stets $w = w\epsilon \in L$ gleichbedeutend mit $w' = w'\epsilon \in L$ ist. Damit gilt $[w] \subseteq L$. Somit erkennt A die Sprache L. Nach dem Theorem von Kleene ist L damit regulär. □

Lemma 2.51 *Es sei L eine reguläre Sprache. Der DEA A sei wie im Beweis von Theorem 2.50 Teil 2 definiert. Dann ist A reduziert. Jeder reduzierte DEA A' mit $L(A') = L$ stimmt mit A bis auf eine Umbenennung der Zustände überein.*

Beweis. Offenkundig ist in $A = (Q, \Sigma, \delta, s, F)$ jeder Zustand $[w]$ vom Start $s = [\epsilon]$ erreichbar, da $\hat{\delta}(s, w) = [w]$ gilt. Enthielte A zwei verschiedene ununterscheidbare Zustände, so würden wir durch eine Reduktion gemäß Lemma 2.47 einen äquivalenten DEA A'' mit echt geringerer Zustandszahl erhalten. Dies widerspricht jedoch der Beobachtung, daß die Zahl der $\sim_L$-Äquivalenzklassen nicht die Zahl der Zustände von A'' übersteigen kann, wie wir im Beweis das Satzes von Nerode gesehen haben. Damit ist A reduziert.

Es sei nun $A' = (Q', \Sigma, \delta', s', F')$ ein reduzierter DEA mit $L(A') = L$. Jeder Zustand aus Q' ist in der Form $\hat{\delta}'(s', w)$ für ein $w \in \Sigma^*$ darstellbar. Wir definieren die Abbildung $F : Q' \to Q$ durch $\hat{\delta}'(s', w) \mapsto [w]$. Die Wohldefiniertheit von F ergibt sich daraus, dass für $\hat{\delta}'(s', w) = \hat{\delta}'(s', w')$ stets $w \sim_L w'$ gilt.

Offenkundig ist F surjektiv. Wir zeigen, daß F injektiv ist: Es gelte $F(q_1) = F(q_2)$ für zwei Zustände $q_1 = \hat{\delta}'(s', v_1)$ und $q_2 = \hat{\delta}'(s', v_2)$ von Q'. Nach Definition von F folgt $[v_1] = [v_2]$. Für jedes $w \in \Sigma$ ergibt sich daher

$$\begin{aligned}
\hat{\delta}'(q_1, w) \in F' &\Leftrightarrow \hat{\delta}'(s', v_1 w) \in F' \\
&\Leftrightarrow v_1 w \in L \text{ (da } L(A') = L) \\
&\Leftrightarrow v_2 w \in L \text{ (da } [v_1] = [v_2]) \\
&\Leftrightarrow \hat{\delta}'(s', v_2 w) \in F' \\
&\Leftrightarrow \hat{\delta}'(q_2, w) \in F'.
\end{aligned}$$

Es sind damit q_1 und q_2 ununterscheidbar, woraus $q_1 = q_2$ folgt.

Es sei nun $q = \hat{\delta}'(s', v) \in Q'$ und $\sigma \in \Sigma$. Wir erhalten

$$F(\delta'(q, \sigma)) \quad = \quad F(\hat{\delta}'(s', v\sigma))$$

$$\begin{aligned} &= [v\sigma] \\ &= \delta([v], \sigma) \\ &= \delta(F(q), \sigma). \end{aligned}$$

Hieraus ergibt sich, daß die Abbildung F homomorph bzgl. σ-Nachfolgern ist. Damit sind A und A' bis auf die Umbenennung F identisch. □

Lemma 2.52 *Es sei L eine reguläre Sprache. Dann stimmen je zwei DEA für L mit minimaler Zustandszahl bis auf eine Umbenennung der Zustände überein.*

Definition 2.53 Es sei L eine reguläre Sprache. Der bis auf Umbenennung der Zustände eindeutig bestimmte DEA A für L mit minimaler Zustandszahl wird der *minimale deterministische endliche Automat für L* genannt.

2.9 Das Pumping-Lemma

Bemerkung 2.54 Jede endliche Sprache L über einem Alphabet Σ ist regulär.

Beweis: 1. Methode: Man kann leicht einen regulären Ausdruck angeben.

2. Methode: Man konstruiert einen DEA A mit $L = L(A)$ wie folgt: Setze

$$Q := \{w \in \Sigma^* \mid w \text{ ist Präfix eines Wortes aus } L\} \cup \{x\}.$$

Hierbei wird das Symbol $x \notin \Sigma$ als Fallenzustand benutzt. Setze $s := \epsilon$, $F := L$, $\delta(w, \sigma) := w\sigma$ falls $w\sigma$ Präfix eines Wortes in L, sonst $\delta(w, \sigma) := x$. Es folgt nach Konstruktion, daß $(s, w) \Longrightarrow_A^* (w, \epsilon)$, falls w Präfix eines Wortes in L, und $(s, w) \Longrightarrow_A^* (x, \epsilon)$ andernfalls.
Für $w \in L$ ist w (als Zustand) final, dh. w wird akzeptiert, alle anderen Worte werden nicht akzeptiert. □

Der angegebene Automat, der die Form eines Baums hat, wird auch *Trie* für die endliche Wortmenge L genannt. Tries werden oft verwendet, um Lexika zu repräsentieren.

Das nachfolgende Lemma stellt eine Standardmethode zur Verfügung, um nachzuweisen, dass eine gegebene Sprache *nicht* regulär ist.

Lemma 2.55 (Pumping- oder Schleifenlemma) *Sei L eine unendliche reguläre Sprache über Σ. Dann gibt es eine Schranke k derart, daß sich jedes Wort $w \in L$ der Länge $|w| > k$ zerlegen läßt in der Form $w = u_1 v u_2$, wo $u_1, v, u_2 \in \Sigma^*$, v nichtleer, so daß $u_1 v^n u_2 \in L$ für jedes $n \in N, n > 0$.*

Beweis: Sei $A = (Q, \Sigma, \delta, s, F)$ ein DEA für L. Sei $k = |Q|$. Sei nun $w = \sigma_1 \cdots \sigma_m$ ein Wort der Länge $m > k$. Setze $q_i := NF_A(s, \sigma_1 \cdots \sigma_i)$ $(0 \leq i \leq m)$. Es ist $q_0, \ldots, q_m$ die Folge der Zustände, die bei der Akzeptanz von w durchlaufen wird. Wegen $m > k$ muß ein Zustand q zweimal durchlaufen werden. Es gelte nun $q_i = q_j$ für ein $i < j$. Wir definieren $u_1 := \sigma_1 \cdots \sigma_i$, $v := \sigma_{i+1} \cdots \sigma_j$ und $u_2 := \sigma_{j+1} \cdots \sigma_m$. Offenkundig werden alle Worte $u_1 v^n u_2$ mit $n \in N$, $n > 0$ in A akzeptiert, da bei der Akzeptanz lediglich die Schleife, die beim Durchlaufen von v in A entsteht, anstatt einmal nun n mal durchlaufen wird. Damit sind alle Wörter $u_1 v^n u_2$ mit $n \in N$, $n > 0$ in L. □

Korollar 2.56 *Sei $\Sigma = \{a, b\}$. Die Sprache L=$\{a^n b^n : n \in N\}$ ist nicht regulär.*

Beweis: Sei $k \in N$. Für jede Zerlegung von $a^k b^k$ in die Form uvw, $w \neq \epsilon$ gilt

1. $v = a^l$, $0 < l \leq k$ oder
2. $v = a^{l_1} b^{l_2}$, $0 < l_1, l_2 \leq k$ oder
3. $v = b^l$, $0 < l \leq k$.

Stets gilt, daß $uv^2w \notin L$. Mit dem Pumping-Lemma ist damit klar, daß L nicht regulär sein kann.

Korollar 2.57 *Deutsch ist nicht regulär.*

Beweis: Betrachte Sätze als Worte, die Menge der Worte als Alphabet Σ.

Betrachte $L_1 = \{$Ein Spion (der einen Spion)^k observiert^l, wird meist selbst oberviert: $k, l \in N\}$.
L_1 ist regulär.
$L_1 \cap Deutsch = L_2 = \{$Ein Spion (der einen Spion)^k observiert^k ,wird meist selbst observiert: $k \in N\}$ ist (, da es nur eine triviale Modifikation von $\{a^n b^n : n \in N\}$ ist) *nicht* regulär.
Wäre Deutsch regulär, dann müßte auch $L_1 \cap Deutsch$ regulär sein, da die regulären Sprachen unter Durchschnitt abgeschlossen sind.

Aufgaben zu Kapitel 2

Aufgabe 2.1 Zeigen Sie, daß für reguläre Ausdrücke α und β stets gilt:

1. $1 \cdot \alpha \equiv \alpha \cdot 1 \equiv \alpha$,
2. $\emptyset \cdot \alpha \equiv \alpha \cdot \emptyset \equiv \emptyset$,
3. $\alpha \cup \beta \equiv \beta \cup \alpha$,
4. $(\alpha \cup \beta) \cdot \gamma \equiv (\alpha \cdot \gamma) \cup (\beta \cdot \gamma)$ und
   $\gamma \cdot (\alpha \cup \beta) \equiv (\gamma \cdot \alpha) \cup (\gamma \cdot \beta)$,
5. $\alpha^* \equiv \emptyset^* \cup \alpha\alpha^*$,
6. $\emptyset^*\alpha \equiv \alpha$

Aufgabe 2.2 Sei $\Sigma = \{a, b\}$. Geben Sie je einen regulären Ausdruck für die folgenden Sprachen an:
(a) $L_1 = \{w \in \Sigma^* :$ w hat kein Vorkommen des Teilworts abba$\}$
(b) $L_2 = \{w \in \Sigma^* :$ w hat genau ein Vorkommen des Teilworts aba$\}$

Aufgabe 2.3 Sei $\Sigma = \{a, b, c\}$. Geben Sie in Form von Zustandsübergangsdiagrammen endliche Automaten an, die folgende Sprachen über Σ akzeptieren.

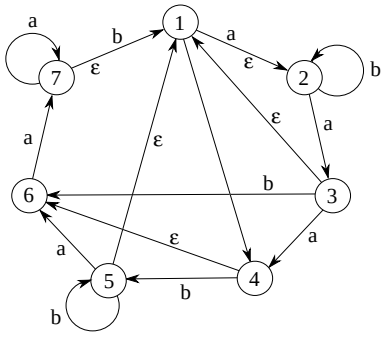
1. Menge aller nichtleeren Wörter, die mit demselben Buchstaben aufhören, mit dem sie auch anfangen.

2. Menge aller Wörter, die zwischen 2 und 4 Vorkommen des Symbols "a" haben.

3. Menge aller Wörter, die *kein* Vorkommen des Teilworts "aba" haben.

4. Menge aller Wörter, wo die Zahl der Vorkommen von "a" und "c" jeweils eine gerade Zahl ist.

Aufgabe 2.4 Gegeben sei der NDEA A aus der nachfolgenden Abbildung. Berechnen Sie für jedes Präfix p des Wortes *aabbabbaaabab*, wie die jeweilige Menge der aktiven Zustände von A nach Konsum des Präfixes p aussieht. Eine naive Heransgehensweise vorausgesetzt, werden Sie werden nach einiger Zeit bemerken, dass sich verschiedene Rechenschritte bei unterschiedlichen Präfixen immer wieder wiederholen. Welche einmalig durchgeführten Hilfsberechnungen können dazu beitragen, die Berechnung der Mengen aktiver Zustände zu vereinfachen? Welche Tabellen werden zum Speichern der Hilfsrechungen benötigt?



Aufgabe 2.5 Der *Levenshtein-Abstand* zweier Wörter v und w ist die minimale Anzahl von Symbollöschungen, Symboleinfügungen und Symbolersetzungen, die man braucht, um v in w umzuschreiben. Beispielsweise haben "sahne" und "sonne" den Levenshtein-Abstand 2 (Ersetzung von a durch o, Ersetzung von h durch n). Geben Sie einen endlichen Automaten A an, der alle Wörter über dem Alphabet $\{a, b, \ldots, z\}$ akzeptiert, die Levenshtein Abstand ≤ 2 zu "sonne" haben.

Aufgabe 2.6 Sei $\Sigma = \{a, b\}$. Geben Sie (in der Form von Zustandsdiagrammen) deterministische endliche Automaten an, die die folgenden Sprachen

akzeptieren:
(a) $L((((aba)^* \cup bb^*a)^*aa))$
(b) $L((b^*aa^*bba(a \cup b)^*))$ und
(c) $\Sigma^* \setminus L((a \cup ba \cup bb)b^*a)^*$

Aufgabe 2.7 Geben Sie reguläre Ausdrücke an, die die Sprache repräsentieren, die durch die deterministischen endlichen Automaten in Abb. 2.8 akzeptiert werden.

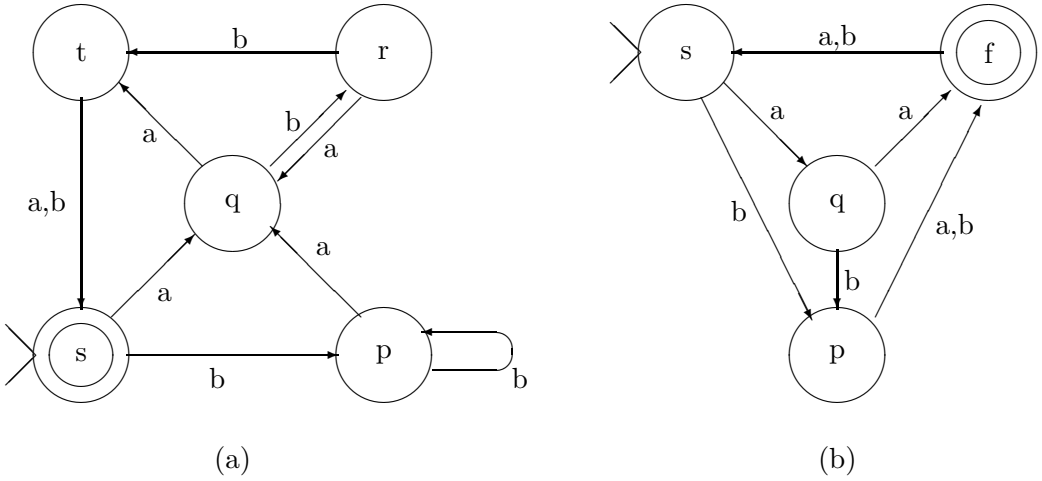


Abbildung 2.8: Wie lauten die regulären Ausdrücke ?

Aufgabe 2.8 Gemäß Definitionen 2.10 und 2.18 haben endliche Automaten – deterministisch oder nicht – nur einen Startzustand, aber i.a. mehrere Finalzustände. Es lassen sich aber leicht *eingeschränkte* endliche Automaten (genau ein Start- und ein Finalzustand) und *verallgemeinerte* endliche Automaten (endlich viele mögliche Start- und Finalzustände) definieren. Welche Änderungen ergeben sich für die Definitionen 2.10, 2.11, 2.18 und 2.19? Welche der folgenden Behauptungen sind richtig (Beweis oder Gegenbeispiel)?

1. zu jedem endlichen Automaten A gibt es einen eingeschränkten endlichen Automaten A' mit $L(A) = L(A')$.

2. zu jedem verallgemeinerten endlichen Automaten A gibt es einen (normalen) endlichen Automaten A' mit $L(A) = L(A')$.

3. zu jedem deterministischen endlichen Automaten A gibt es einen eingeschränkten deterministischen endlichen Automaten A' mit $L(A) = L(A')$.

4. zu jedem verallgemeinerten deterministischen endlichen Automaten A gibt es einen (normalen) deterministischen endlichen Automaten A' mit $L(A) = L(A')$.

Aufgabe 2.9 Beweisen Sie Lemma 2.27 (Hinweis: zeige $\subseteq$ und $\supseteq$).

Aufgabe 2.10 Es sei $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.
(a) Der reguläre Ausdruck $\alpha := 0 \cup (1 \cup 2 \cup \cdots \cup 9)(0 \cup 1 \cup \cdots \cup 9)^*$ repräsentiert die Menge aller Dezimaldarstellungen natürlicher Zahlen. Geben Sie möglichst einfache endliche Automaten A_1, A_2 und A_3 an, die $L(0)$, $L(1 \cup 2 \cup \cdots \cup 9)$ und $L(0 \cup 1 \cup \cdots \cup 9)$ akzeptieren. Folgen Sie nun der Konstruktion des Beweises von Satz 2.38, um einen endlichen Automaten A für $L(\alpha)$ zu erhalten. Transformieren Sie dann A in einen äquivalenten DEA.
(b) Geben Sie direkt einen EA A_{GD} an, der genau die Dezimaldarstellungen *gerader* natürlicher Zahlen akzeptiert. (Machen Sie diesen nicht unnötig groß, drei Zuständen reichen!) Transformieren Sie A_{GD} in einen äquivalenten DEA.
(c) Geben Sie einen DEA an, der genau die Dezimaldarstellungen von durch drei teilbaren natürlichen Zahlen akzeptiert. Zur Erleichterung können Sie zunächst einen EA konstruieren, der beliebige Ziffernfolgen mit durch drei teilbarer Quersumme akzeptiert – drei Zustände reichen hierfür.

Aufgabe 2.11 Zeigen Sie: ist A ein DEA, so wird die in Lemma 2.46 definierte Folge $\sim_0, \sim_1, \cdots$ spätestens nach $n - 2$ Schritten konstant, dh. es stimmen $\sim_{n-2}$ und $\sim_{n-1}$ überein. Geben Sie für jede natürliche Zahl $n > 2$ einen DEA A_n mit n Zuständen an, für den tatsächlich $\sim_{n-3}$ und $\sim_{n-2}$ noch verschieden sind.

Aufgabe 2.12 Es sei ein EA A mit Eingabealphabet Σ gegeben. Das Wort $w = \sigma_1 \cdots \sigma_k \in \Sigma^*$ *kann zu einem Wort in* $L(A)$ *aufgebläht werden*, wenn

es Worte $w_0, \cdots, w_k$ in Σ^* gibt derart, daß $w_0\sigma_1 w_1 \cdots \sigma_k w_k \in L(A)$. Zeigen Sie: es ist für jeden EA A und jedes Wort w entscheidbar, ob w zu einem Wort in $L(A)$ aufgebläht werden kann.

Aufgabe 2.13 In Abb. 2.9 sehen Sie Gerüste für Automaten über dem Alphabet $\{a, h\}$. Das linke "Gerüst" steht für einen DEA H und das rechte für einen DEA A. Ergänzen Sie die Diagramme derart, daß sich durch Reduktion des resultierenden DEA H der DEA A ergibt. Tragen Sie dazu Start- und Finalzustände in die Diagramme ein, ebenso Labels und Richtungen der angedeuteten Übergänge, sowie, falls nötig, zusätzliche Übergänge (die jedoch nur zwischen schon verbundenen Zuständen einzurichten sind, damit die H- bzw. A-Form nicht verloren geht). Wenn Sie Lust haben, können Sie auch noch versuchen Ergänzungen anzugeben, so daß sich das H zu der Form X oder I reduziert.

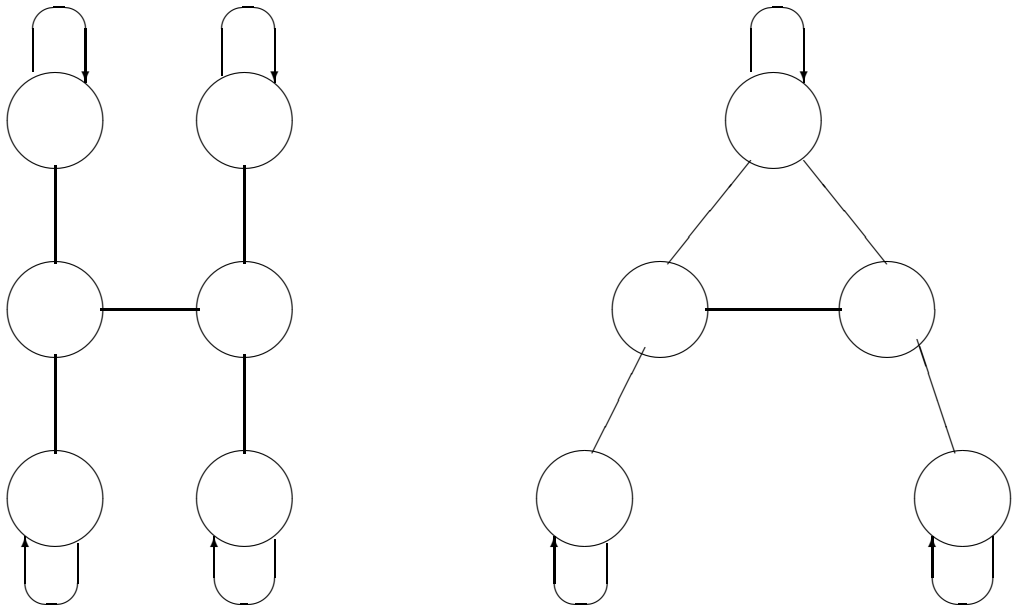


Abbildung 2.9: Automatengerüste

Aufgabe 2.14 Beweisen Sie Lemma 2.52.

Aufgabe 2.15 Es sei Σ ein Alphabet. Ein ω-Wort über Σ ist eine unendliche Folge $\sigma_1\sigma_2\sigma_3\cdots$ von Symbolen aus Σ (formaler können ω-Wörter als

Abbildungen von N in Σ beschrieben werden). Hintergrund dieser Aufgabe ist die einstmals von Axel Thue gestellte Frage, ob es möglich ist, ein ω-Wort über Σ zu finden, das quadratfrei ist, dh. kein Teilwort der Form vv hat, wo $v \in \Sigma^+$ nichtleer ist. Es ist recht trickreich zu zeigen, daß dies in der Tat im Fall $|\Sigma| > 2$ möglich ist. (Dieses Ergebnis dürfen Sie für (b) verwenden.)
(a) Zeigen Sie, daß für $|\Sigma| = 2$ kein Wort w der Länge $|w| \geq 4$ quadratfrei ist.
(b) Zeigen Sie, daß es für $|\Sigma| > 2$ unendlich viele quadratfreie Wörter über Σ gibt, und daß es sogar unendlich viele quadratfreie ω-Wörter über Σ gibt.
(c) Verwenden Sie (b), das Pumping-Lemma und eine kleine Zusatzidee, um zu zeigen, daß für $|\Sigma| > 2$ die Sprache $L = \{w \in \Sigma^* \mid w$ hat ein Teilwort der Form $vv, v \in \Sigma^+\}$ nicht regulär ist.

Aufgabe 2.16 Konstruieren Sie unter Verwendung des im Kapitel beschriebenen Verfahrens einen regulären Ausdruck, der die Sprache repräsentiert, die durch den DEA in Abb. 2.10 akzeptiert wird.

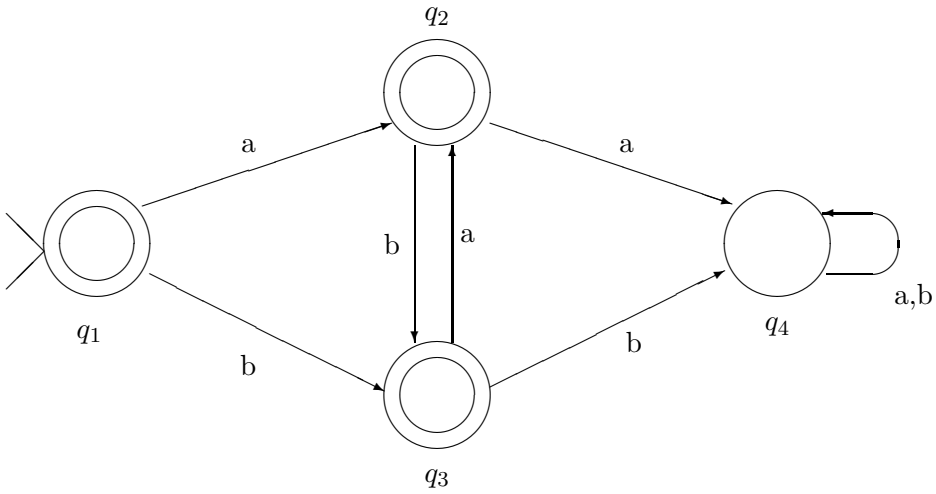


Abbildung 2.10: Regulärer Ausdruck ?

Aufgabe 2.17 Es sei a *kein* Element des Alphabets Σ und L_1 eine beliebige Sprache über Σ, die zumindest ein nichtleeres Wort enthält. Zeigen Sie (etwa durch eine kleine Modifikation des Beweises des Pumping-Lemmas), daß die Sprache $L_2 = \{a^n w a^n : \ w \in L_1, n \in N\}$ nicht regulär ist. (Dies läßt sich so interpretieren, daß reguläre Sprachen kein „Zählgedächtnis“ haben.)

Kapitel 3

Die Chomsky-Hierarchie

Definition 3.1 Ein *Semi-Thue-System* ist ein Paar (Σ, Π), wo Σ ein Alphabet ist und $\Pi = \{l_i \longrightarrow r_i;\ \text{i=1,}\cdots\text{,n}\} \subseteq \Sigma^* \times \Sigma^*$ ein Produktionensystem ($n \geq 0$). Seien $u, v \in \Sigma^*$: u wird vermöge Π in einem Schritt nach v abgeleitet, in Zeichen $u \longrightarrow^1_\Pi v$, falls es eine Zerlegung $u = w_1 l w_2$, $w_1, w_2 \in \Sigma^*$, gibt und eine Regel $l \longrightarrow r$ in Π wo $v = w_1 r w_2$.
Mit $\longrightarrow^+_\Pi$ bzw. $\longrightarrow^*_\Pi$ wird die transitive bzw. reflexiv transitive Hülle von $\longrightarrow^1_\Pi$ bezeichnet. ($\longrightarrow^*_\Pi$ Ableitbarkeit)

Beispiel 3.2 $S = (\{a, b\}, \{aa \to b,\ \ ab \to ba\})$

$$\begin{array}{rcl} b\underline{aa}bab & \longrightarrow^1_\Pi & b\underline{b}bab \\ baab\underline{ab} & \longrightarrow^1_\Pi & baabba \end{array}$$

Definition 3.3 [Grammatik] Eine *(Phrasenstruktur-)Grammatik* ist ein Quadrupel G=(N,T,Π,S), wobei gilt:

- N ist ein Alphabet (Nichtterminale),
- T ist ein Alphabet (Terminale), wobei $T \cap N = \emptyset$; $\Sigma = N \cup T$,
- (Σ, Π) ist ein Semi-Thue-System
- $S \in N$ ist Startsymbol

Die *von G erzeugte Sprache* ist L(G)=$\{w \in T^* : S \longrightarrow^* w\}$ wobei $\longrightarrow^*$ die durch Π bestimmte Ableitungsrelation ist. Man schreibt manchmal auch $\longrightarrow^*_G$.

Notation: Wir schreiben für $l \longrightarrow r_1,\ \cdots,\ l \longrightarrow r_k$ auch $l \longrightarrow r_1 \mid r_2 \mid \cdots \mid r_k$

Beispiel 3.4 1. $G = (\{Z\}, \{(,)\}, \{Z \to \epsilon \mid ZZ \mid (Z)\}, Z)$ liefert Klammergerüste wohlgeklammerter Ausdrücke.
$Z \to ZZ \to Z(Z) \to Z((Z)) \to Z(()) \to (Z)(()) \to ()(())$
(Aufgabe: Ist L(G) regulär?)

2. $G = (\{S, B, C\}, \{a, b, c\}, \Pi, S)$ wobei Π folgende Übergänge hat:

   1: $S \longrightarrow aSBC$

   2: $S \longrightarrow aBC$

   3: $CB \longrightarrow BC$

   4: $aB \longrightarrow ab$

   5: $bB \longrightarrow bb$

   6: $bC \longrightarrow bc$

   7: $cC \longrightarrow cc$

   Es gilt $L(G) = \{a^n b^n c^n :\ n \in N_+\}$. (L(G) ist nach dem Pumping-Lemma nicht regulär.)

   $L(G) \supseteq \{a^n b^n c^n :\ n \in N_+\}$**:** einfach

   $L(G) \subseteq \{a^n b^n c^n :\ n \in N_+\}$**:** da in den Regeln 4–7 nur Nichtterminale durch entsprechende Terminale ersetzt werden, existiert zu jeder Herleitung $S \longrightarrow^*_G w,\ w \in T^*$ eine ähnliche, in der alle Regelanwendungen vom Typ 1–3 vor denen vom Typ 4–7 stattfinden.

   $$S \overset{1-3}{\longrightarrow}{}^*_G \quad x \overset{4-7}{\longrightarrow}{}^*_G \quad w$$

   Leicht zu sehen: x hat die Form $a^k y$ für ei $y \in \{B, C\}^*$ und $k > 0$.
   Wegen der Gestalt der Regeln 4–7 müssen Nichtterminale von links nach rechts durch Terminale ersetzt werden. Hätte y ein Teilwort CB, so würde der Prozeß stoppen. Daraus folgt $L(G) \subseteq \{a^{n_1} b^{n_2} c^{n_3} :\ n_i > 0\}$. Der Rest ist einfach.

Noam Chomsky stellte (in den 50er Jahren) folgende Einteilung von Grammatiken auf: ($\Sigma = N \cup T$)

Definition 3.5 [Chomsky-Hierarchie] Für i=0,1,2,3 hat G=(N,T,Π,S) den Typ i, falls G die i-te der folgenden Restriktionen erfüllt:

0: keine Restriktion

1: jede Produktion in Π hat die Form $x_1 A x_2 \longrightarrow x_1 y x_2$ wobei $x_1, x_2 \in \Sigma^*$, $y \in \Sigma^+$, $A \in N$ mit der möglichen Ausnahme der Regel $S \longrightarrow \epsilon$, deren Existenz aber impliziert, daß S nicht in einer rechten Seite einer Regel aus Π auftritt.

2: jede Produktion in Π hat die Form $A \longrightarrow x$ wobei $a \in N$ und $x \in \Sigma^*$.

3: jede Produktion in Π hat die Form $A \longrightarrow aB$ oder A$\longrightarrow \epsilon$ wobei $A, B \in N$ und $a \in T$.

Eine *Sprache* L hat den Typ i, falls L von einer Grammatik vom Typ i erzeugt wird. Grammatiken vom Typ

- 1 heißen kontext-sensitiv
- 2 heißen kontextfrei
- 3 heißen rechtslinear

Eine Grammatik heißt linkslinear, falls alle Produktionen die Form $A \longrightarrow Ba$ oder $A \longrightarrow \epsilon$ (entspr. Typ 3) haben.
Produktionen der Form $A \longrightarrow a$ $(A \in N, a \in T)$ heißen terminierend, l$\longrightarrow \epsilon$ heißen ϵ- oder leere Produktion und $l \longrightarrow r$ wobei $\mid l \mid \leq \mid r \mid$ heißen nichtverkürzt oder beschränkt.

VORSICHT: Eine Sprache kann von Grammatiken verschiedenen Typs erzeugt werden und damit mehrere Typen haben. Man kann sich durchaus einen Typberg vorstellen, auf dessen Spitze Typ 3 steht.

Theorem 3.6 *Eine Sprache $L \subseteq T^*$ ist regulär $\Longleftrightarrow$ L wird von einer rechtslinearen Grammatik erzeugt.*

Der Vollständigkeit halber sollen hier noch folgende Entsprechungen erwähnt werden:

- Typ 0 $\longleftrightarrow$ Turing-Maschinen
- Typ 1 $\longleftrightarrow$ linear beschränkte Automaten
- Typ 2 $\longleftrightarrow$ Kellerautomaten
- Typ 3 $\longleftrightarrow$ endliche Automaten

Aufgaben zu Kapitel 3

Aufgabe 3.1 (a) Ein Wort, das von vorne gelesen dasselbe ergibt wie von hinten gelesen, heißt Palindrom. Schreiben Sie eine kontextfreie Grammatik, die genau die Palindrome über dem Alphabet $\{a, b\}$ erzeugt. (Handelt es sich um eine reguläre Sprache?)
(b) Schreiben Sie eine kontextsensitive Grammatik, die die Sprache $\{a^n b^n a^n b^n : \ n \in N\}$ erzeugt.

Aufgabe 3.2 Eine Grammatik $G = (N, T, \Pi, S)$ heißt *linear*, wenn alle Produktionen in Π die Form $A \to uBv$ oder $A \to \epsilon$ haben, wo $u, v \in T^*$, $A, B \in N$. Geben Sie ein Beispiel einer linearen Grammatik G, wo L(G) nicht regulär ist.

Aufgabe 3.3 Geben Sie eine Sprache L vom Typ 3 an und für jedes $i = 3, 2, 1, 0$ eine Grammatik G_i, die genau L erzeugt.

Aufgabe 3.4 (a) Es sei L eine reguläre Sprache. Beweisen Sie, daß dann auch die Sprache $L^{rev} = \{u^{rev} : \ u \in L\}$ regulär ist.
(b) Zeigen Sie, daß die Klasse der regulären Sprachen genau diejenigen Sprachen umfaßt, die durch eine rechtslineare Grammatik erzeugt werden.
(c) Verwenden Sie (a), um zu zeigen, daß die Klasse der regulären Sprachen genau diejenigen Sprachen umfaßt, die durch eine linkslineare Grammatik erzeugt werden.

Kapitel 4

Kontextfreie Spachen, Kellerautomaten

4.1 Kellerautomaten

Definition 4.1 Ein *Kellerautomat* ist ein Sechstupel $A = (Q, I, \Gamma, \Delta, s, F)$. Dabei sind

- Q eine endliche Zustandsmenge,
- I ein (Eingabe-) Alphabet,
- Γ ein (Stack-) Alphabet,
- $\Delta \subseteq (Q \times I^* \times \Gamma^*) \times (Q \times \Gamma^*)$ eine Übergangsrelation,
- $s \in Q$ der Startzustand,
- $F \subseteq Q$ die Menge der Finalzustände.

Eine *Konfiguration* von A ist ein Element aus $Q \times I^* \times \Gamma^*$ und besteht intuitiv aus einem aktuellen Zustand, einem Restwort und einer Stack- bzw. Kellerbeschreibung. Die Konfiguration $(q, w, \beta\gamma)$ kann *direkt von* $(p, vw, \alpha\gamma)$ *erreicht werden* genau dann, wenn es einen Übergang $((p, v, \alpha), (q, \beta))$ gibt; wir schreiben dann $(p, vw, \alpha\gamma) \Longrightarrow_A^1 (q, w, \beta\gamma)$.
$\Longrightarrow_A^*$ bezeichne wieder die reflexive transitive Hülle von $\Longrightarrow_A^1$.

Ein Wort $w \in I^*$ wird von A *akzeptiert* (wir schreiben $w \in L(A)$) genau dann, wenn $(s, w, \epsilon) \Longrightarrow_A^* (f, \epsilon, \epsilon)$ für ein $f \in F$.

Eine Grundidee (bzw. häufige Verwendungsart) des Stacks bzw. Kellers besteht darin, Regeln $A \longrightarrow wBx$ einer kontextfreien Grammatik ($w \in T^*, B \in N, x \in \Sigma^*$) algorithmisch in der Form „von Zustand A lies w, gehe zu B und vermerke x als noch zu erledigen auf dem Stack“ zu interpretieren. Man vergleiche dies mit regulären Sprachen bzw. rechtslinearen Grammatiken. Dort kann eine Regel $A \longrightarrow aB$ algorithmisch als „von A lies a und gehe zu B“ interpretiert werden.

Beispiel 4.2 Wir betrachten den Kellerautomaten $A = (\{s, q, f\}, \{a, b\}, \{a, b, \sharp\}, \Delta, s, \{f\})$ mit der Übergangsrelation Δ mit den Regeln

1. $((s, \epsilon, \epsilon), (q, \sharp))$
2. $((q, a, \sharp), (q, a\sharp))$
3. $((q, b, \sharp), (q, b\sharp))$
4. $((q, a, b), (q, \epsilon))$
5. $((q, b, a), (q, \epsilon))$
6. $((q, a, a), (q, aa))$
7. $((q, b, b), (q, bb))$
8. $((q, \epsilon, \sharp), (f, \epsilon))$

In diesem Fall wird der Keller dazu genutzt, den momentanen Überschuss eingegangener Symbole a bzw. b zu notieren. Es ist

$$L(A) = \{w \in \{a, b\}^* \mid w \text{ enthält gleich viele } a\text{'s wie } b\text{'s}\}.$$

Als Beispiel für einen Akzeptanzvorgang betrachten wir die Eingabe $w = abba$. Sie führt zu folgender Konfigurationsfolge:

$$\begin{aligned} (s, abba, \epsilon) &\Longrightarrow_A^1 (q, abba, \sharp) \\ &\Longrightarrow_A^1 (q, bba, a\sharp) \end{aligned}$$

$$\begin{aligned} &\Longrightarrow_A^1 \quad (q, ba, \sharp) \\ &\Longrightarrow_A^1 \quad (q, a, b\sharp) \\ &\Longrightarrow \quad (q, \epsilon, \sharp) \\ &\Longrightarrow_A^1 \quad (f, \epsilon, \epsilon) \end{aligned}$$

Da $f \in F$ gilt, wird w von A akzeptiert.

4.2 Äquivalenz von kontextfreienGrammatiken und Kellerautomaten

Das erste große Ziel ist es, zu zeigen, daß die kontextfreien Sprachen genau diejenigen Sprachen sind, die durch einen Kellerautomaten erkannt werden können.

Notation: Im folgenden Abschnitt seien jeweils

- $u, v, w, u_1, \cdots$ Worte im Eingabealphabet,
- $\alpha, \beta, \gamma, \alpha_1, \cdots$ Worte im Stackalphabet,
- $x, y, z, x_1, \cdots$ Worte über $\Sigma = N \cup T$ oder Stackworte,
- p, q, r, s Zustände
- N, M nicht-terminale Symbole.

Definition 4.3 Sei $G = (N, T, \Pi, S)$ eine kontextfreie Grammatik (kfG), $\Sigma = N \cup T$. Eine Herleitung $S \longrightarrow_G^* x$ heißt *leftmost (lm)*, wenn in jedem Schritt das am weitesten links stehende nicht-terminale Symbol ersetzt wurde.

Beispiel 4.4 G habe die Produktionsregeln $S \longrightarrow ABC$, $A \longrightarrow a$, $B \longrightarrow b$, $C \longrightarrow c$.

$$S \longrightarrow^1 ABC \longrightarrow^1 aBC \longrightarrow^1 abC \longrightarrow abc$$

ist eine leftmost-Herleitung.

$$S \longrightarrow^1 ABC \longrightarrow^1 AbC \longrightarrow^1 abC \longrightarrow abc$$

ist keine leftmost-Herleitung.

Lemma 4.5 *Sei $G = (N, T, \Pi, S)$ kfG. Falls $S \longrightarrow_G^* w$, so existiert eine lm-Herleitung $S \overset{lm}{\longrightarrow}_G^* w$.*

Beweis: Falls eine Herleitung $S \longrightarrow w$ im Schritt i nicht leftmost: ändere
$S \longrightarrow^1 \cdots \longrightarrow^1 w_1NxMy \overset{i}{\longrightarrow}^1 w_1Nxzy \longrightarrow^1 \cdots \longrightarrow w_1Nz_1 \overset{j}{\longrightarrow} w_1x_1z_1 \longrightarrow \cdots \longrightarrow w$
in $S \longrightarrow^1 \cdots \longrightarrow^1 w_1NxMy \longrightarrow^1 w_1x_1xMy \longrightarrow w_1x_1x_1y \longrightarrow \cdots$ und lassen j-ten Schritt nun weg.
Man erhält so eine Herleitung von w mit größerem leftmost-Teil am Anfang. Durch Iteration erhält man eine leftmost-Herleitung für w.

Lemma 4.6 *Sei $A = (Q, I, \Gamma, \Delta, s, F)$ ein Kellerautomat. Falls $(p, w, \alpha) \Longrightarrow_A^* (q, \epsilon, \beta)$ und $\gamma \in \Gamma^*$, so auch $(p, w, \alpha\gamma) \Longrightarrow_A^* (q, \epsilon, \beta\gamma)$.*

Beweis: triviale Induktion.

Theorem 4.7 *Zu jeder kontextfreien Grammatik $G = (N, T, \Pi, S)$ existiert ein Kellerautomat A mit $L(A) = L(G)$.*

Beweis: Konstruktion eines Kellerautomaten:
Setze $A = (Q, T, \Sigma, \Delta, s, F)$ wo $\Sigma := N \cup T$, $Q := \{s, q\}$, $F := \{q\}$ und wo Δ folgende Übergänge hat:

1. Stack-Initialisierung: $((s, \epsilon, \epsilon), (q, S))$,
2. Vorhersagen: $((q, \epsilon, M), (q, x))$ für jede Regel $M \longrightarrow x$ *in* Π.
3. Eingabevergleich: $((q, a, a), (q, \epsilon))$ für jedes $a \in T$

Zunächst benötigen wir noch folgende **Hilfsbehauptungen:**
1) Falls $S \overset{lm}{\longrightarrow}_G^* w\alpha$, $\alpha \in N\Sigma^* \cup \{\epsilon\}$, dann $(q, w, S) \Longrightarrow_A^* (q, \epsilon, \alpha)$.
2) Falls $(q, w, S) \Longrightarrow_A^* (q, \epsilon, \alpha)$, $\alpha \in \Sigma^*$, so $S \longrightarrow_G^* w\alpha$.

Der Rest ist dann einfach: für $w \in T^*$ gilt

$$w \in L(G) \iff S \longrightarrow_G^* w$$

$$\begin{aligned} &\iff S \overset{lm}{\longrightarrow}{}^*_G w \\ &\iff (q, w, S) \Longrightarrow^*_A (q, \epsilon, \epsilon) \\ &\iff w \in L(A). \end{aligned}$$

Beweis der Hilfsbehauptungen:
1) durch Induktion über Herleitungslänge k.
Induktionsanfang: im Falle $k = 0$ folgt $w = \epsilon$, $\alpha = S$, es gilt wie gefordert $(q, \epsilon, S) \Longrightarrow^*_A (q, \epsilon, S)$.
Induktionsschritt: es sei $k \geq 0$. Wir betrachten eine leftmost Herleitung mit $k + 1$ Schritten

$$S \to^1_G \cdots \to^1_G w_k | M_k \alpha_k \to^1_G w_k x \alpha_k = w_k v | \alpha.$$

Der Strich "|" deutet hierbei lediglich das Ende des maximalen terminalen Präfixes des hergeleiteten Strings an. Im letzten Ableitungsschritt wurde die Regel $M_k \to x$ verwendet, $w = w_k v$ ist das maximale terminale Präfix, α das verbleibende Suffix von $w_k x \alpha_k = w\alpha$. Nach Induktionsvoraussetzung gilt $(q, w_k, S) \Longrightarrow^*_A (q, \epsilon, M_k \alpha_k)$. Hieraus folgt $(q, w, S) \Longrightarrow^*_A (q, v, M_k \alpha_k)$. Eine Vorhersage führt zur Konfiguration $(q, v, x\alpha_k) = (q, v, v\alpha)$. Mittels Eingabevergleiche erreichen wir in A die Konfiguration (q, ϵ, α).

2) Induktion über die Zahl $k \geq 0$ der beteiligten Konfigurationsübergänge.
Induktionsanfang: $k = 0$. Dann gilt $w = \epsilon$ und $\alpha = S$. Natürlich gilt $S \longrightarrow^*_G S$.
Induktionsschritt: Es sei $k \geq 0$. Der Akzeptanzdurchlauf in A führe von (q, w, S) in k Schritten zu $(q, v, \alpha_1\gamma)$, danach in einem Schritt zu (q, ϵ, α). Hierbei sei $((q, v, \alpha_1), (q, \beta))$ der verwendete Übergang von A, es gilt dann $\alpha = \beta\gamma$. Hat w die Form $w = w_k v$, so folgt $(q, w_k, S) \Longrightarrow^*_A (q, \epsilon, \alpha_1\gamma)$. Aus der Induktionsvoraussetzung ergibt sich $S \longrightarrow^*_G w_k \alpha_1 \gamma$.

- Falls $((q, v, \alpha_1), (q, \beta))$ eine Vorhersage mit zugehöriger Produktionsregel $M \longrightarrow \beta$ ist, so folgt $v = \epsilon$, $w = w_k$ sowie $\alpha_1 = M$. Wir erhalten $S \longrightarrow^*_G wM\gamma \longrightarrow^1_G w\beta\gamma = w\alpha$.
- Falls $((q, v, \alpha_1), (q, \beta))$ ein Eingabevergleich ist, so folgt $v = a \in T$, w hat die Form $w_k a$, es gilt $\alpha_1 = a$, $\beta = \epsilon$ und $\alpha = \gamma$. Wir erhalten $S \longrightarrow^*_G w_k a\alpha = w\alpha$.

Dies beendet den Beweis.

Bemerkung 4.8 Im Beweis wurde ein Top-Down-Kellerautomat für die Grammatik konstruiert, d.h. die Akzeptanz entspricht der Konstruktion eines Parse-Baums von der Spitze S bis zu den terminalen Blättern. Man kann auch einen Bottom-Up-Kellerautomaten zu der Grammatik G angeben. Hierzu setzt man $A = (\{s, q\}, T, N \cup T, \Delta, s, \{q\})$ wo Δ folgende Übergänge hat:

1. Shift: $((s, a, \epsilon), (s, a))$ für jedes $a \in T$,
2. Reduce: $((s, \epsilon, x^{rev}), (s, M))$ für jede Regel $M \longrightarrow x \ \ in \ \ \Pi$,
3. Finish: $((s, \epsilon, S), (q, \epsilon))$.

Dann gilt wieder $L(A) = L(G)$. Wir verzichten auf den Beweis.

Für die Umkehrung des Lemmas brauchen wir noch folgende Definition.

Definition 4.9 Ein Kellerautomat $A = (Q, I, \Gamma, \Delta, s, F)$ heißt *einfach*, wenn gilt:

1. wenn $((q, u, \alpha), (p, \beta)) \in \Delta$, so gilt $\mid \alpha \mid \leq 1$,
2. wenn $((q, u, \epsilon), (p, \beta)) \in \Delta$, so auch $((q, u, M), (p, \beta M)) \in \Delta$, für jedes $M \in \Gamma$.

Lemma 4.10 *Falls L von einem Kellerautomaten A akzeptiert wird, so auch von einem einfachen Kellerautomaten A'.*

Beweis: Falls A nicht einfach, so machen wir folgende Änderungen:
Zu (1): Falls $((q, u, A_1 \cdots A_k), (p, \beta)) \in \Delta$, so füge neue Zustände $t_1, \cdots, t_{k-1}$ zu Q hinzu und ersetze $((q, u, A_1 \cdots A_k), (p, \beta))$ durch $((q, \epsilon, A_1), (t_1, \epsilon))$, $((t_1, \epsilon, A_2), (t_2, \epsilon))$, $\cdots, ((t_{k-2}, \epsilon, A_{k-1}), (t_{k-1}, \epsilon))$, $((t_{k-1}, u, A_k), (p, \beta))$. Das Resultat dieser Änderungen sei die Übergangsrelation Δ'.
Zu (2): füge zu Δ' die nötigen Übergänge $((q, u, M), (p, \beta M))$ hinzu. Dies ändert an der akzeptierten Sprache nichts.

Theorem 4.11 *Falls die Sprache L von einem Kellerautomaten A akzeptiert wird, so ist L kontextfrei.*

Beweis: Die nachfolgende Konstruktion einer kontextfreien Grammatik wird manchmal als „Tripelkonstruktion“ bezeichnet. Sei $A = (Q, I, \Gamma, \Delta, s, F)$ OBdA einfach. Sei $G := (N, I, \Pi, S)$ wobei $N := \{S\} \cup \{\langle p, B, q\rangle \mid p, q \in Q, B \in \Gamma \cup \{\epsilon\}\}$. Intuitiv repräsentiert ein nichtterminales Symbol $\langle p, B, q\rangle$ einen terminalen String, der in A bei der Wanderung von p nach q konsumiert wird, wobei auf dem Stack im Resultat B entfernt wurde (hierbei können über dem betreffenden Vorkommen von B zusätzliche Symbole auf- und abgebaut werden). Die Relation Π hat folgende Regeln:

1. $S \longrightarrow \langle s, \epsilon, f\rangle$ für alle $f \in F$,
2. $\langle p, B, q\rangle \longrightarrow u\ \langle r, B_1, q_1\rangle\ \langle q_1, B_2, q_2\rangle \cdots \langle q_{n-1}, B_n, q\rangle$
   für jeden Übergang $((p, u, B), (r, B_1 \cdots B_n)) \in \Delta$, wo $n > 0$, $B \in \Gamma \cup \{\epsilon\}$ und für alle $q_1, \cdots, q_{n-1}, q \in Q$.
   (Lies: Eine Wanderung in A von p nach q, bei der im Resultat B gepopt wird, kann dadurch zustande kommen, daß man von p zunächst nach r läuft, wobei u konsumiert wird und B durch $B_1 \cdots B_n$ ersetzt wird, und dann n Wanderungen der Form $\langle q_i, B_{i+1}, q_{i+1}\rangle$ durchführt, die zu q führen.)
3. $\langle p, B, q\rangle \longrightarrow u\ \langle r, \epsilon, q\rangle$
   für alle $((p, u, B), (r, \epsilon)) \in \Delta$ mit $B \in \Gamma \cup \{\epsilon\}$ und für alle $q \in Q$,
4. $\langle p, \epsilon, p\rangle \longrightarrow \epsilon$ für jedes $p \in Q$.

Behauptung: Für alle $p, q \in Q$, $B \in \Gamma \cup \{\epsilon\}$, $w \in I^*$:

$$\langle p, B, q\rangle \longrightarrow_G^* w \iff (p, w, B) \Longrightarrow_A^* (q, \epsilon, \epsilon).$$

Der Rest ist dann einfach: $w \in L(G) \iff \langle s, \epsilon, f\rangle \longrightarrow_G^* w$ für ein $f \in F \iff (s, w, \epsilon) \Longrightarrow_A^* (f, \epsilon, \epsilon)$ für ein $f \in F \iff w \in L(A)$.

Beweis der Behauptung:
„$\Longrightarrow$“. Sei $\langle p, B, q\rangle \longrightarrow_G^* w$ eine Herleitung in k Schritten. Wir verwenden Induktion über k. Gilt $k = 1$, so wurde eine Regel vom Typ (4) angewandt, damit gilt $p = q$ und $B = \epsilon = w$. Es sei nun $k > 1$. Dann muß der erste Herleitungsschritt vom Typ (2) oder (3) sein.

1. Falls eine Regel vom Typ (2) korrespondierend zum Automatenübergang $((p, u, B), (r, B_1 \cdots B_n)) \in \Delta$ angewandt wurde, so hat die Herleitung die Form

   $$\langle p, B, q\rangle \longrightarrow_G^1 u\ \langle r_1, B_1, q_1\rangle \cdots \langle q_{i-1}, B_i, q_i\rangle \cdots \langle q_{n-1}, B_n, q\rangle \longrightarrow_G^* w.$$

Es habe hierbei w die Form $uv_1 \dots v_n$ wo $\langle q_{i-1}, B_i, q_i\rangle \longrightarrow_G^* v_i$ die betreffenden Teilherleitungen der obigen Herleitung sind ($i = 1, \dots, n$, mit $q_0 := r$). Gemäß Induktionsvoraussetzung gilt $(q_{i-1}, v_i, B_i) \Longrightarrow_A^* (q_i, \epsilon, \epsilon)$ für $1 \leq i \leq n$.

$$\begin{aligned}(p, w, B) = (p, uv_1 \cdots v_n, B) &\Longrightarrow_A^1 (r, v_1 \cdots v_n, B_1 \cdots B_n)\\ &\Longrightarrow_A^* (q_1, v_2 \cdots v_n, B_2 \cdots B_n)\\ &\cdots\\ &\Longrightarrow_A^* (q_n, \epsilon, \epsilon).\end{aligned}$$

2. Falls eine Regel vom Typ (3) korrespondierend zur Automatenübergangsregel $((p, u, B), (r, \epsilon))$ angewandt wurde, so hat die Herleitung die Form
$$\langle p, B, q\rangle \longrightarrow_G^1 u \, \langle r, \epsilon, q\rangle \longrightarrow^* uv = w.$$
Nach Induktionsvoraussetzung gilt $(r, v, \epsilon) \Longrightarrow_A^* (q, \epsilon, \epsilon)$. Damit folgt unter Verwendung der Übergangsregel
$$(p, uv, B) \Longrightarrow_A^1 (r, v, \epsilon) \Longrightarrow_A^* (q, \epsilon, \epsilon).$$

„$\Longleftarrow$“. Sei $(p, w, B) \Longrightarrow_A^* (q, \epsilon, \epsilon)$ eine Konfigurationsabfolge von A mit k Schritten. Wir verwenden Induktion über k. Falls $k = 0$ gilt, so folgt $w = \epsilon = B$, $p = q$, und wir haben $\langle p, \epsilon, p\rangle \longrightarrow_G^* \epsilon$ durch Anwenden einer Regel vom Typ 4. Es gelte nun $k > 0$. Die Konfigurationsfolge hat die Form
$$(p, w, B) \Longrightarrow_A^1 (r, v, B_1 \cdots B_n) \Longrightarrow_A^* (q, \epsilon, \epsilon).$$
Hier gilt $w = uv$ wo u beim ersten Übergang konsumiert wird. Wir analysieren die Form der beim ersten Übergang verwendeten Regel aus Δ. Falls $B = \epsilon$, so hat die Regel die Form $((p, u, B), (r, B_1 \cdots B_n))$. Falls $B \in \Gamma$, so hat die Regel die Form $((p, u, B), (r, B_1 \cdots B_n))$ oder es gilt $B = B_n$ und die Regel hat die Form $((p, u, \epsilon), (r, B_1 \cdots B_{n-1}))$. Da aber A einfach ist, können wir auch in diesem Fall annehmen, dass die Regel $((p, u, B), (r, B_1 \cdots B_n))$ verwendet wurde. Wir unterscheiden nun zwei Fälle

1. Falls $n > 0$. Da A einfach ist, werden die B_i in der zweiten Hälfte der Konfigurationsfolge sukzessive abgebaut. Es gibt Zustände $q_1, \cdots, q_{n-1}$ und Teilwörter $v_1, \cdots, v_n$ von $v = v_1 \cdots v_n$ mit

$$\begin{aligned}(r, v_1 \cdots v_n, B_1 \cdots B_n) &\Longrightarrow_A^* (q_1, v_2 \cdots v_n, B_2 \cdots B_n)\\ &\cdots\\ &\Longrightarrow_A^* (q_n, \epsilon, \epsilon) = (q, \epsilon, \epsilon).\end{aligned}$$

Aufgrund der Form der verwendeten Automatenregel $((p, u, B), (r, B_1 \cdots B_n))$ haben wir in Π die Produktionsregel

$$\langle p, B, q\rangle \longrightarrow u\ \langle r, B_1, q_1\rangle\ \langle q_1, B_2, q_2\rangle \cdots \langle q_{n-1}, B_n, q\rangle$$

vom Typ 2 in Π, wobei nach Induktionsvoraussetzung $\langle r, B_1, q_1\rangle \longrightarrow_G^* v_1$, $\cdots$, $\langle q_{n-1}, B_n, q\rangle \longrightarrow_G^* v_n$. Also gilt

$$\langle p, B, q\rangle \longrightarrow_G^* uv_1 \cdots v_n = uv = w.$$

2. Falls $n = 0$. Es hat die Konfigurationsfolge die Form

$$(p, w, B) \Longrightarrow_A^1 (r, v, \epsilon) \Longrightarrow_A^* (q, \epsilon, \epsilon).$$

Wir haben die Regel $\langle p, B, q\rangle \longrightarrow u\ \langle r, \epsilon, q\rangle$ vom Typ 3. Nach Induktionsvoraussetzung gilt $\langle r, \epsilon, q\rangle \longrightarrow_G^* v$. Es folgt $\langle p, B, q\rangle \longrightarrow_G^* uv = w$.

4.3 Abschlußeigenschaften kontextfreier Sprachen

Theorem 4.12 *Die Klasse der kontextfreien Sprachen ist abgeschlossen unter*

1. *Vereinigung,*
2. *Konkatenation und*
3. *Kleene-Stern.*

Beweis: Seien $G_i = (N_i, T_i, \Pi_i, S_i)$ (i=1,2) kontextfreie Grammatiken; OBdA gelte $N_1 \cap N_2 = \emptyset$.

1. Setze $G = (N_1 \cup N_2 \cup \{S\}, T_1 \cup T_2, \Pi_1 \cup \Pi_2 \cup \{S \longrightarrow S_1 \mid S_2\}, S)$, wobei S ein neues Startsymbol ist. Dann gilt $L(G) = L(G_1) \cup L(G_2)$, G ist kontextfrei.
2. Setze $G = (N_1 \cup N_2 \cup \{S\}, T_1 \cup T_2, \Pi_1 \cup \Pi_2 \cup \{S \longrightarrow S_1S_2\}, S)$, wobei S ein neues Startsymbol ist. Dann gilt $L(G) = L(G_1) \circ L(G_2)$, G ist kontextfrei.

3. Setze $G = (N_1 \cup \{S\}, T_1, \Pi_1 \cup \{S \longrightarrow \epsilon \mid SS \mid S_1\}, S)$ wo S ein neues Symbol ist. Es ist klar, daß G kontextfrei ist.
   $L(G) \subseteq L(G_1)^*$: man zeigt, daß jede G-Herleitung eines nichtleeren Wortes w durch Abänderung der Reihenfolge der Expansionen so umgeformt werden kann, daß zuerst $k \geq 0$ Anwendungen der Regel $S \longrightarrow SS$ erfolgen, dann k Schritte der Form $S \to S_1$ und danach nur Regeln aus Π_1. Damit ist $w \in L(G_1)^*$.
   $L(G_1)^* \subseteq L(G)$: trivial

Bemerkung 4.13 Der Durchschnitt zweier kontextfreier Sprachen ist nicht notwendig kontextfrei; das Komplement einer kontextfreien Sprache ist ebenfalls nicht notwendig kontextfrei. Dies werden wir weiter unten sehen.

Lemma 4.14 *Der Durchschnitt einer kontextfreien Sprache mit einer regulären Sprache ist stets kontextfrei.*

Beweis: Wir können annehmen, daß wir einen Kellerautomaten A_1 für L_1 und einen DEA A_2 für L_2 haben.
$A_1 = (Q_1, I_1, \Gamma_1, \Delta_1, s_1, F_1)$, $A_2 = (Q_2, I_2, \delta_2, s_2, F_2)$.
Setze $A = (Q_1 \times Q_2, I_1 \cap I_2, \Gamma_1, \Delta, (s_1, s_2), F_1 \times F_2)$ wobei $(((p_1, p_2), w, \alpha), ((q_1, q_2), \beta))) \in \Delta :\Longleftrightarrow ((p_1, w, \alpha), (q_1, \beta)) \in \Delta_1 \bigwedge (p_2, w) \Longrightarrow_{A_2}^* (q_2, \epsilon)$
Durch eine einfache Induktion zeigt man, daß
$((s_1, s_2), w, \epsilon) \Longrightarrow_A^* ((q_1, q_2), \epsilon, \beta) \Longleftrightarrow (s_1, w, \epsilon) \Longrightarrow_{A_1}^* (q_1, \epsilon, \beta) \bigwedge (s_2, w) \Longrightarrow_{A_2}^* (q_2, \epsilon)$ $\forall w \in (I_1 \cap I_2)^*$.
Der Rest ist trivial.

4.4 Parse-Bäume

Definition 4.15 Sei G=(N, T, Π, S) kfG. Die Menge der (G-)*Parse-Bäume* ist wie folgt definiert.

1. Für jedes $A \in N \cup T$ ist A ein Parse-Baum mit Wurzel A und Blattfolge A.

2. Sind $T_1, \cdots, T_n$ Parse-Bäume mit Wurzeln $A_1, \cdots, A_n$ und Blattfolgen Bl(1),$\cdots$,Bl(n) und ist $A \longrightarrow A_1 \cdots A_n \in \Pi$, so ist der Baum mit Wur-

zel A und unmittelbaren Teilbäumen (von links nach rechts) $T_1, \cdots, T_n$ ein Parsebaum mit Wurzel A. Die Blattfolge ist Bl(1),$\cdots$,Bl(n).

3. Ist $A \longrightarrow \epsilon \in \Pi$, so ist der Baum mit Wurzel A und einzigem Blatt ϵ ein Parse-Baum mit Wurzel A und Blatt(folge) ϵ.

4. Jeder Parse-Baum wird nach den oberen drei Regeln mittels endlich vieler Regelanwendungen gebildet.

Ein Parse-Baum T heißt *vollständig*, falls jedes Blatt terminal ist und falls T Wurzel S hat. T heißt *Parse-Baum für* $\sigma_1 \cdots \sigma_k$, wenn $\sigma_1 \cdots \sigma_k \in T$ die Blätter von T sind, von links nach rechts, ϵ auslassend.

Notation: Ich verwende der leichteren Schreibweise wegen die Termnotation für Bäume. Einen Baum mit Wurzel A und Nachfolgern b, c, d und e schreibe ich $A(b, c, d, e)$.

Bemerkung: Es gilt $\sigma_1 \cdots \sigma_k \in L(G) \iff$ es existiert ein vollständiger Parse-Baum für $\sigma_1 \cdots \sigma_k$.

Beispiel: $S \longrightarrow Ae \mid aCe \mid aD$, $A \longrightarrow Bd$, $B \longrightarrow abc$, $C \longrightarrow bcd$, $D \longrightarrow bE$, $E \longrightarrow cde$.
(Kleinbuchstaben sind terminal, Großbuchstaben nicht.)
S(A(B(a,b,c),d),e), S(a,C(b,c,d),e) und S(a,D(b,E(c,d,e))) sind Parse-Bäume für abcde. Sie zeigen, daß die gewählte Grammatik *mehrdeutig* ist, dh. mehrere leftmost-Herleitungen eines Wortes zuläßt.

Bemerkung 4.16

1. Ein vollständiger Parse-Baum faßt alle Herleitungen eines terminalen Worts zusammen, die sich nur durch die Reihenfolge der Regelanwendungen unterscheiden.
   Beispiel: $\Pi : S \longrightarrow AB$, $A \longrightarrow a$, $B \longrightarrow b$
   Die Herleitungen $S \longrightarrow AB \longrightarrow aB \longrightarrow ab$ und $S \longrightarrow AB \longrightarrow Ab \longrightarrow ab$ haben beide den Parse-Baum S(A(a),B(b)).
   Es ist ein wesentlicher Unterschied, ob ein Wort mehrere Parse-Bäume hat oder nur mehrere Herleitungen.

2. Wir geben hier keine exakte Definition eines Baumes (vergleiche Logikbücher)

3. Das Konzept des Parse-Baumes ist genau auf die Eigenschaft der Kontextfreiheit abgestimmt, wo in Regeln links nur ein Symbol steht.

4. Die Parse-Bäume für rechtslineare Grammatiken haben Kammform: S(a,B(b,C(c,D(d,...)))) - sie wachsen nach *rechts*.
   Beachte: Hier entspricht ein Parse-Baum genau einer möglichen Herleitung.

Definition 4.17 Die *Höhe h(T)* eines Parse-Baumes T der kontextfreien Grammatik G ist die maximale Länge eines Pfades von T. Ein Pfad ist eine Folge von Knoten derart, daß jeder direkte Nachfolger (in der Folge) ein direkter unterer Nachbar im Baum ist.

Beispiel 4.18 $h(S(a, X(b, c))) = 2$.

Theorem 4.19 (Pumping-Theorem für kontextfreie Sprachen) *Sei G eine kontextfreie Grammatik. Dann existiert ein $k \in N$ derart, daß jeder String $w \in L(G)$ der Länge $|w| > k$ dargestellt werden kann in der Form $w = u_1v_1u_2v_2u_3$, wo $v_1v_2 \neq \epsilon$ und wo jedes Wort der Form $u_1v_1^nu_2v_2^nu_3$ ($n \in N$) zu L(G) gehört.*

Beweis: Sei $G = (N, T, \Pi, S)$. Sei $p = max\{|x| \mid \text{es gibt eine Regel } A \longrightarrow x \in \Pi\}$. Setze $k = p^{|N|}$. Jeder Parse-Baum T mit $h(T) = r$ hat höchstens p^r Blätter. Also: Hat T mehr als k Blätter, so gilt $h(T) > |N|$, d.h. ein $A \in N$ muß in einem Pfad (mindestens) zwei Mal auftreten. Ist $w \in L(G)$, $|w| > k$, so gilt dasselbe für jeden vollständigen Parse-Baum für w. Sei $|w| > k$. Wir betrachten einen vollständigen Parsebaum für w minimaler Höhe, unter diesen einen Parsebaum minimaler Knotenzahl. In einem Pfad trete etwa $A \in N$ (zumindest) zweimal auf. Der Baum hat die in Abbildung 4.1 dargestellte Form. Es gilt $v_1v_2 \neq \epsilon$, denn sonst könnte man den Teilbaum T (vgl. Abbildung) mit Wurzel A direkt am oberen A anhängen und man erhielte einen kleineren, nichthöheren Parse-Baum für w.

Um die im Lemma beschriebenen Wörter zu erfasssen, wiederholen wir den durch die beiden dargestellten Vorkommen von A bestimmten "Kontext" (Baum mit Loch) n mal und hängen unten wieder T ein. Dadurch erhalten wir einen vollständigen Parse-Baum für $u_1v_1^nu_2v_2^nu_2 \in L(G)$. □

Bemerkung 4.20 Wie schon im Fall der regulären Sprachen wird auch hier das Pumping-Theorem benutzt, um zu zeigen, daß bestimmte Sprachen *nicht* kontextfrei sind.

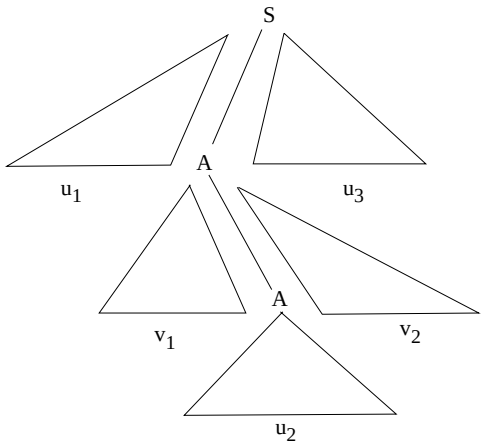


Abbildung 4.1: Illustration zum Beweis des Pumpinglemmas für kontextfreie Sprachen

Beispiel 4.21 Die Sprache $L = \{a^n b^n c^n \mid n \in N\}$ ist nicht kontextfrei.

Beweis: Wir nehmen an, L sei kontextfrei. Dann gibt es nach dem Pumping-Theorem ein m so, daß $a^m b^m c^m$ in der angegebenen Weise zerlegt werden kann. Sowohl in v_1 als auch in v_2 kann nur ein Buchstabentyp auftreten. Damit existiert ein Buchstabentyp, der weder in v_1 noch in v_2 auftritt. Der wäre dann aber in $u_1 v_1^n u_2 v_2^n u_3$ unterrepräsentiert. WIDERSPRUCH!

Korollar 4.22 *Die Klasse der kontextfreien Sprachen ist weder unter Durchschnitts- noch unter Komplementbildung abgeschlossen.*

Beweis: $L_1 = \{a^n b^n c^m : \ m, n \geq 0\}$, $L_2 = \{a^m b^n c^n : \ m, n \geq 0\}$ sind kontextfrei, denn mit $L_{10} = \{a^n b^n : n \geq 0\}$, $L_{11} = \{c^m : m \geq 0\}$, $L_{20} = \{a^m : m \geq 0\}$ *und* $L_{21} = \{b^n c^n : n \geq 0\}$, die offenbar kontextfrei sind, gilt dies auch für $L_1 = L_{10} \circ L_{11}$ *und* $L_2 = L_{20} \circ L_{21}$.
Aber: $L_1 \cap L_2 = T^* \setminus ((T^* \setminus L_1) \cup (T^* \setminus L_2))$ ist nicht kontextfrei !

4.5 Kontextfeie Grammatiken in Chomsky-Normalform

Definition 4.23 Zwei Grammatiken G und G' heißen (schwach) äquivalent genau dann, wenn $L(G) = L(G')$.

Definition 4.24 Eine kontextfreie Grammatik $G = (N, T, \Pi, S)$ heißt ϵ-*frei*, wenn sie keine Regel der Form $A \longrightarrow \epsilon$ enthält, mit der möglichen Ausnahme $S \longrightarrow \epsilon$, falls $\epsilon \in L(G)$. Dann soll S nicht in rechten Seiten einer Ableitungsregel vorkommen.

Lemma 4.25 *Zu jeder kontextfreien Grammatik $G = (N, T, \Pi, S)$ läßt sich algorithmisch eine äquivalente ϵ-freie kontextfreie Grammatik G' konstruieren.*

Beweis: Setze

$$\begin{aligned} W_1 &:= \{A \in N : A \longrightarrow \epsilon \in \Pi\}, \\ W_{i+1} &:= \{A \in N : A \longrightarrow x \in \Pi, x \in W_i^*\} \quad (i \geq 1). \end{aligned}$$

Dann gilt

1. $W_i \subseteq W_{i+1}$ für alle $i \geq 1$,
2. Falls $W_i = W_{i+1}$, so $W_i = W_{i+k}$ $(k \geq 1)$,
3. $W_n = W_{n+1}$, wobei n=$\mid N \mid$,
4. $W_n = \{A \in N : A \longrightarrow_G^* \epsilon\}$.

Setze

$$\begin{aligned} N' &:= N \cup \{S'\}, \text{ wobei } S' \text{ ein neues Startsymbol ist,} \\ T' &:= T \\ \Pi' &:= \{S' \longrightarrow S\} \cup \mathit{Red}(\Pi) \cup \Pi_\epsilon. \end{aligned}$$

Hierbei sei $\Pi_\epsilon := \emptyset$, falls $\epsilon \notin L(G)$ und $\Pi_\epsilon := \{S' \longrightarrow \epsilon\}$ sonst. Weiter steht $\mathit{Red}(\Pi)$ für die Menge aller Regeln, die sich aus Regeln von Π dadurch

ergeben, daß man auf der rechten Regelseite irgendwelche (z.B. alle, einige oder keine) nichtterminale Symbole aus W_n wegläßt, ohne daß jedoch die rechte Seite hierbei leer werden darf. Es entstehen aus einer Regel von Π ggfs. mehrere Regeln von $Red(\Pi)$.

Klar: G' ist ϵ-frei. Ist $w \neq \epsilon$ in G herleitbar, $w \in T^*$, so auch in G': dazu läßt man bei allen Anwendungen der Regeln auf der rechten Seite alle Nichtterminale weg, die später zu ϵ terminalisiert werden. Die modifizierten Regeln sind in Π', also erhalten wir eine G'-Herleitung. Umgekehrt kann jede G'-Herleitung durch Auffüllen der rechten Regelseiten zu einer G-Herleitung umgeschrieben werden.

Definition 4.26 [Kettenproduktion] Sei $G = (N, T, \Pi, S)$ eine kontextfreie Grammatik. Eine Produktion der Form $A \longrightarrow B$ mit $A, B \in N$ heißt *Kettenproduktion.*

Lemma 4.27 *Zu jeder ϵ-freien kontextfreien Grammatik G läßt sich algorithmisch eine äquivalente ϵ-freie kontextfreie Grammatik G' ohne Kettenproduktionen konstruieren.*

Beweis: Für alle $A \in N$ definiere

$$\begin{aligned} W_0(A) &:= \{A\}, \\ W_{i+1}(A) &:= \{B \in N : C \longrightarrow B \in \Pi \text{ für ein } C \in W_i(A)\} \cup W_i(A). \end{aligned}$$

Dann gelten

1. $W_i(A) \subseteq W_{i+1}(A)$
2. falls $W_i(A) = W_{i+1}(A)$, so gilt $W_i(A) = W_{i+k}(A)$ $(k \geq 1)$
3. $W_n(A) = W_{n+1}(A)$, wobei n=$\mid N \mid$
4. $W_n(A) = \{B \in N : A \longrightarrow_G^* B\}$. (Dies gilt, da G ϵ-frei.)

Wir definieren $G' := (N', T', \Pi', S')$ wo

$$\begin{aligned} N' &:= N, \\ T' &:= T, \\ S' &:= S, \\ \Pi' &:= \{A \longrightarrow x \mid x \in (N \cup T)^* \setminus N, \exists B \in W_n(A) : B \longrightarrow x \in \Pi\}. \end{aligned}$$

G' enthält keine Kettenproduktionen und ist ϵ-frei. Eine G-Herleitung kann durch ,,Zusammenziehen" von G-Kettenproduktionen zu einer G'-Herleitung umgeformt werden. Umgekehrt kann jede G'-Herleitung in eine G-Herleitung umgeformt werden, da Regeln aus Π' durch mehrere Regeln aus Π imitiert werden können.

Definition 4.28 [terminalisierbar] Sei $G = (N, T, \Pi, S)$ kontextfreie Grammatik. $A \in N$ heißt *terminalisierbar* genau dann, wenn es ein $w \in T^*$ gibt mit $A \longrightarrow_G^* w$.

Lemma 4.29 *Für jede kontextfreie Grammatik G ist entscheidbar, ob das Startsymbol S terminalisierbar ist. S ist terminalisierbar genau dann, wenn $L(G) \neq \emptyset$. Zu jeder kontextfreien Grammatik G mit $L(G) \neq \emptyset$ läßt sich algorithmisch eine äquivalente kontextfreie Grammatik G' konstruieren, wo jedes nichtterminale Symbol terminalisierbar ist. Falls G eine ϵ-freie Grammatik ist und frei von Kettenproduktionen, so gilt dies auch für G'.*

Beweis: Definiere

$$\begin{aligned} W_1 &:= \{A \in N : \exists w \in T^* : A \longrightarrow w \in \Pi\} \\ W_{k+1} &:= \{A \in N \mid \exists x \in (W_k \cup T)^* : A \longrightarrow x \in \Pi\} \cup W_k. \end{aligned}$$

W_k enthält somit alle Nichtterminale, aus denen ein terminales Wort mittels eines Ableitungsbaumes der Höhe $\leq k$ ableitbar ist. Es gilt

1. $W_k \subseteq W_{k+1}$,
2. falls $W_k = W_{k+1}$, so gilt $W_k = W_{k+m}$ (m$\geq$1),
3. $W_n = W_{n+1}$, wobei n=$|N|$
4. $W_n = \{A \in N : A \longrightarrow_G^* w$ für ein $w \in T^*\}$.

Die berechenbare Menge W_n enthält also alle terminalisierbaren Nichtterminale. Offenkundig gilt, daß S terminalisierbar ist genau dann, wenn $L(G) \neq \emptyset$. Im folgenden setzen wir voraus, daß $L(G) \neq \emptyset$. Wir definieren $G' := (N', T', \Pi', S')$ wo

$$\begin{aligned} N' &:= W_n, \\ T' &:= T, \\ S' &:= S, \\ \Pi' &:= \{A \longrightarrow x \in \Pi \mid \ A, x \in (T' \cup N')^*\}. \end{aligned}$$

Da $\Pi' \subseteq \Pi$ folgt $L(G') \subseteq L(G)$. Falls andererseits $S \longrightarrow_G^* w \in T^*$, so sind alle Nichtterminale der Ableitung terminalisierbar. Nach Punkt 4 und der Wahl von N' gilt dies auch in G'. Daher sind alle Ableitungsregeln dieser Ableitung in Π' und damit gilt $S \longrightarrow_{G'}^* w$.

Nachfolgend sei für die betrachteten Grammatiken G stets $L(G)$ nichtleer.

Definition 4.30 [erreichbar] Sei $G = (N, T, \Pi, S)$ kontextfreie Grammatik und $A, B \in N \cup T$. Das Symbol $B \in N \cup T$ heißt *von A aus erreichbar* (wo $A \in N$) genau dann, wenn es $x, y \in (N \cup T)^*$ gibt mit $A \longrightarrow_G^* xBy$.

Lemma 4.31 *Zu jeder kontextfreien Grammatik G läßt sich algorithmisch eine äquivalente kontextfreie Grammatik G' konstruieren, so daß jedes Symbol aus $N' \cup T'$ von S' aus erreichbar ist. Ist G eine ϵ-freie Grammatik, frei von Kettenproduktionen und ist jedes Nichtterminal in G terminalisierbar, so gilt dies alles auch für G'.*

Beweis: Definiere

$$\begin{aligned} W_1 &:= \{S\}, \\ W_{k+1} &:= \{B \in N \cup T \mid \exists x, y \in (N \cup T)^*, A \in W_k : A \longrightarrow xBy \in \Pi\} \cup W_k \end{aligned}$$

W_k umfaßt alle Symbole (Terminale und Nichtterminale) von G, die von S aus mit weniger als k Ableitungsschritten erreichbar sind. Es gelten

1. $W_k(S) \subseteq W_{k+1}(S)$,
2. falls $W_k(S) = W_{k+1}(S)$, so $W_k(S) = W_{k+m}(S)$ $(m \geq 1)$,
3. $W_n(S) = W_{n+1}(S)$, wobei n=$|N|$,
4. $W_n(S) = \{A \in (N \cup T) : \exists x, y \in (N \cup T)^* \text{ mit } S \longrightarrow xAy\}$

Setze $G' := (N', T', \Pi', S')$ wo

$$\begin{aligned} N' &:= N \cap W_n(S), \\ T' &:= T \cap W_n(S), \\ S' &:= S, \\ \Pi' &:= \{A \longrightarrow x \in \Pi \mid A \in N'\}. \end{aligned}$$

Offenkundig gilt $L(G') \subseteq L(G)$. Ist $w \in T^*$ in G ableitbar, so sind alle Symbole von S aus erreichbar und somit $w \in L(G')$. G' ist ϵ-frei und frei von Kettenproduktionen, sofern G es ist.

Sei jedes $A \in N$ in G terminalisierbar. Für $A \in N$ sei $deg_G(A) := min\{k|A \in W_k\}$, W_k wie im Terminalisierungsalgorithmus (vgl. Beweis von Lemma 4.29). Angenommen

$$M := \{A \in N' \mid A \text{ in } G' \text{ nicht terminalisierbar}\} \neq \emptyset.$$

Wähle aus M ein Element minimalen Grades $deg_G(A) = m$ (es gilt $m > 0$). Da A in G terminalisierbar ist, existiert eine Regel $A \longrightarrow x \in \Pi$, wo jedes Symbol aus x einen kleineren Grad als A hat. Da A in G von S erreichbar, gilt dies auch für alle Nichtterminale in x. Solche muß es geben, andernfalls $A \longrightarrow x \in \Pi$', A in G' terminalisierbar. Nichtterminale B aus x haben kleineren Grad, sind also in M. WIDERSPRUCH!.

Definition 4.32 [reduziert] Eine kontextfreie Grammatik G heißt *reduziert* genau dann, wenn $\Pi = \emptyset$ oder falls für jedes $A \in N \cup T$ Worte $x, y \in (N \cup T)^*$, $w \in T^*$ existieren mit $S \longrightarrow_G^* xAy \longrightarrow_G^* w$.

Korollar: Zu jeder kontextfreien Grammatik G läßt sich algorithmisch eine äquivalente reduzierte kontextfreie Grammatik G' konstruieren, die ϵ-frei und frei von Kettenproduktionen ist.

Definition 4.33 [Chomsky-Normalform] Eine ϵ-freie reduzierte kontextfreie Grammatik $G = (N, T, \Pi, S)$ ist in *Chomsky-Normalform* genau dann, wenn alle Regeln in Π die Form $A \longrightarrow a$ $(A \in N,\ a \in T)$, $A \longrightarrow BC$ $(A, B, C \in N)$ oder $S \longrightarrow \epsilon$ haben.

Theorem 4.34 *Zu jeder kontextfreien Grammatik $G = (N, T, \Pi, S)$ läßt sich algorithmisch eine äquivalente kontextfreie Grammatik G' in Chomsky-Normalform konstruieren.*

Beweis: G sei ObdA ϵ-frei, reduziert und frei von Kettenproduktionen. Sei $T = \{a_1, \cdots, a_k\}$. Falls a_i in der Regel $B \longrightarrow xa_iy$ vorkommt, wo $x, y \in (N \cup T)^+$, so führen wir ein neues nichtterminales Symbol A_i ein, eine Regel $A_i \longrightarrow a_i$, ersetzen $B \longrightarrow xa_iy$ durch $B \longrightarrow xA_iy$. Durch Iteration kann

man alle diese Fälle eliminieren. Danach sind alle Regeln von der Form $S \longrightarrow \epsilon$, $B \longrightarrow a_i$ oder $B \longrightarrow B_1 \cdots B_n$, mit $B, B_1, \cdots, B_n, S \in N,\ a_i \in T,\ k \geq 2$. Falls $k > 2$, nimm neue Symbole $N_1, \cdots, N_{k-2}$ und ersetze $B \longrightarrow B_1 \cdots B_n$ durch $B \longrightarrow B_1 N_1$, $N_1 \longrightarrow B_2 N_2$, $\cdots$, $N_{k-2} \longrightarrow B_{k-1} B_k$.

4.6 Der Algorithmus von Younger, Cocke, Kasami

Gegeben sei eine kontextfreie Grammatik G in Chomsky-Normalform. Der folgende *Algorithmus von Younger, Cocke und Kasami* entscheidet für jedes $w \in T^*$, ob $w \in L(G)$ oder nicht:

Schritt 0: falls $w = \epsilon$, so gilt $w \in L(G) \Longleftrightarrow S \longrightarrow \epsilon \in \Pi$.
Schritt 1: falls $w = \sigma_1 \cdots \sigma_n$, $n \geq 1$, so berechne das obere Dreieck einer $n \times n$-Matrix $M_{i,j}(w)$ wie folgt:
a) Hauptdiagonale (Bottom) für $i = 1, \ldots, n$

$$M_{i,i} = \{A \in N : A \longrightarrow \sigma_i \in \Pi\}$$

b) l-te Diagonale (Up) für $l = 2, \cdots, n$: für $j = l, \cdots, n$:

$$(i := j - l + 1;\ M_{i,j} := \bigcup_{i \leq k < j} \{A : A \longrightarrow BC \in \Pi, B \in M_{i,k}, C \in M_{k+1,j}\})$$

(Kommentar: die horizontalen Nummern j der l-ten Diagonale laufen von l bis n. Die zugehörigen vertikalen Nummern sind durch $i = j - l + 1$ gegeben. Die Summation bei der Berechnung von $M_{i,j}$ beginnt immer mit den B-Einträgen der Hauptdiagonalen derselben Höhe i, also bei $M_{i,i}$, und geht bis zu $M_{i,j-1}$. Die Positionen der Partnerfelder ergeben sich zu $M_{k+1,j}$.)

Es gilt $w \in L(G) \iff S \in M_{1,n}(w)$

Beispiel: G habe die Produktionen

1. $S \longrightarrow AB$
2. $A \longrightarrow C1$
3. $A \longrightarrow C2$
4. $B \longrightarrow 1C$

5. $B \longrightarrow 2C$
6. $2 \longrightarrow DD$
7. $C \longrightarrow c$
8. $D \longrightarrow d$
9. $1 \longrightarrow d$

a) zu testen: liegt cdc in L(G) ?
b) zu testen: liegt cdddc in L(G) ?

Theorem 4.35 *Der Algorithmus von Younger, Cocke und Kasami ist korrekt und vollständig.*

Beweis: Zeige allgemeiner: für jedes $A \in N,\ \epsilon \neq w \in T^*$:
$A \in M_{1,|w|}(w) \Longleftrightarrow A \longrightarrow_G^* w$
Beweis durch Induktion über $\mid w \mid$:
Induktionsanfang: $\mid w \mid = 1 :\ A \longrightarrow_G^* w \Longleftrightarrow A \longrightarrow w \in \Pi \Longleftrightarrow A \in M_{1,1}(w)$
Induktionsannahme: Es gelte die Behauptung bis zur Wortlänge $n-1$.
Induktionsschritt: Sei $\mid w \mid = n \geq 2$:
$A \in M_{1,|w|}(w) \Longleftrightarrow (\exists u \neq \epsilon \neq v \in T^*,\ w = uv,\ A \longrightarrow BC \in \Pi)\ und\ B \in M_{1,|w|}(w)\ und\ C \in M_{1+|u|,|w|}(w) \Longleftrightarrow (\exists u \neq \epsilon \neq v \in T^*,\ w = uv,\ A \longrightarrow BC \in \Pi)\ und\ B \in M_{1,|u|}(u)\ und\ C \in M_{1,|v|}(v) \overset{IV}{\Longleftrightarrow} (\exists u \neq \epsilon \neq v \in T^*,\ w = uv,\ A \longrightarrow BC \in \Pi)\ und\ B \longrightarrow_G^* u\ und\ C \longrightarrow_G^* v \Longleftrightarrow A \longrightarrow_G^* w$.

Bemerkung 4.36 (n=Wortlänge) Wenn man die Größe von G als konstant ansieht, so sind für jeden Matrixeintrag $k \leq n$ Rechenschritte nötig. Insgesamt erhält man einen Zeitbedarf t_{YCK} der Größenordnung $O(n^3)$, da die Größe der Matrix $O(n^2)$ hat.

4.7 Der Earley-Algorithmus

Der Earley-Algorithmus dient zur Entscheidung der Akzeptanz bei beliebiger gegebener kontextfreier Grammatik $G = (N, T, \Pi, S)$. Wir

führen eine neue Wurzel (die Superwurzel) $\overline{S}, \overline{S} \notin N$, ein Stopzeichen $\sharp \notin N \cup T$ und eine Startproduktion $\overline{S} \longrightarrow S\sharp$ ein. Dann setzen wir $\hat{\Pi} = \Pi \cup \{\overline{S} \longrightarrow S\sharp\}, \hat{N} = N \cup \{\overline{S}\}, \hat{T} = T \cup \{\sharp\}$.

Definition 4.37 Eine *gepunktete Produktion* (zu G) hat die Form $X \longrightarrow X_1 \cdots X_i \bullet X_{i+1} \cdots X_p$ wobei $0 \leq i \leq p$ und X$\longrightarrow X_1 \cdots X_p \in \Pi$ gilt.

Definition 4.38 Um im Earley-Algorithmus zu testen, ob $a_1 \cdots a_n \in$L(G) gilt, wird $a_1 \cdots a_n\sharp$ als Eingabe verwendet. Ein *Zustand* q des Earley-Algorithmus ist eine gepunktete Produktion, die mit einer Startnummer nach dem Pfeil versehen ist:

$$q : X \longrightarrow j \mid \alpha \bullet \beta$$

hierbei gilt also $X \longrightarrow \alpha\beta \in \hat{\Pi}$. Der Zustand q heißt *final* genau dann, wenn $\beta = \epsilon$. Zustand q heißt *terminal* genau dann, wenn $\beta = a\nu$ mit $a \in \hat{T}$,

$\nu \in (\hat{T} \cup \hat{N})^*$, q heißt *nicht-terminal*, falls $\beta = A\nu$ mit $A \in \hat{N}, \nu \in (\hat{T} \cup \hat{N})^*$.

Eine *Zustandsfolge* ist eine geordnete Menge von Zuständen. Wir werden bei Eingabe von $a_1 \cdots a_n\sharp$ nacheinander die Zustandsfolgen $Q_0, \cdots, Q_{n+1}$ berechnen. Ist $X \longrightarrow j \mid \alpha \bullet \beta \in Q_i$, dann gilt $j \leq i \leq n+1$ und wir schreiben auch $X \longrightarrow j \mid \alpha \overset{i}{\bullet} \beta$, lies ,,Aufgrund der erfolgten Kontrolle von $a_1 \cdots a_j a_{j+1} \cdots a_i$ ist möglich, daß X für die Herleitung eines Teilwortes $a_{j+1} \cdots a_i \cdots a_l$ $(l \geq i)$ verwendbar ist.“ Insbesondere liest sich gegebenenfalls $\overline{S} \longrightarrow 0 \mid S\sharp \overset{n+1}{\bullet}$ so: ,,nach vollständiger Kontrolle von $a_1 \cdots a_n\sharp$ steht fest, daß aus $\overline{S}$ tatsächlich $a_1 \cdots a_n\sharp$ herleitbar ist.“

Definition 4.39 [Earley-Algorithmus]

Initialisierung: Setze $Q_i = \emptyset$ für $0 \leq i \leq n+1$, i=0 und füge $\overline{S} \longrightarrow 0 \mid \bullet S\sharp$ zu Q_0.

Hüllenbildung, Entscheidung: Bearbeite die Zustandsfolgen Q_i der Reihe nach (nummeriere alle Produktionen beim Bearbeiten und vermerke, aus welchen Regeln sie entstanden sind). Bilde jeweils den Abschluß bezüglich Vorhersage, Rekonstrukion und Eingabevergleich.

Falls Q_i abgeschlossen ist, $i < n+1$, berechne Q_{i+1}. (Die Berechnung von Q_{i+1} beginnt eigentlich schon beim Eingabevergleich in Q_i.) Falls am Ende $Q_{n+1} = \{\overline{S} \longrightarrow 0 \mid S\sharp\bullet\}$, so akzeptiere das Eingabewort, andernfalls verwerfe es.

Vorhersage: Falls Q_i einen nichtterminalen Zustand $X \longrightarrow j \mid \alpha \bullet A\nu$ enthält, so füge alle Produktionen $A \longrightarrow \gamma \in \Pi$ als Zustand $A \longrightarrow i \mid \bullet\gamma$ zu Q_i.

Rekonstruktion: Falls Q_i einen finalen Zustand $X \longrightarrow w \mid \alpha\bullet$ enthält und Q_w einen Zustand $C \longrightarrow g \mid \delta \bullet X\nu$, so füge den Zustand $C \longrightarrow g \mid \delta X \bullet \nu$ zu Q_i. (Anschaulich: Alle Produktionen, für die in Q_w eine Vorausschau für X gemacht wurde, werden jetzt in Q_i vermerkt mit dem Unterschied, daß der Punkt hinter dem X steht.)

Eingabevergleich: Falls Q_i einen terminalen Zustand $X \longrightarrow j \mid \alpha \bullet a\nu$ enthält und falls a_{i+1}=a gilt, so füge $X \longrightarrow j \mid \alpha a \bullet \nu$ zu Q_{i+1}.

Beispiel 4.40 G habe die Superwurzel S und folgende Produktionen:

1. $S \longrightarrow R\sharp$
2. $R \longrightarrow aRb$
3. $R \longrightarrow A$
4. $A \longrightarrow Aa$
5. $A \longrightarrow a$

Es soll geprüft werden, ob G das Wort $aaab\sharp$ erzeugt.
$Q_0 = \{$
01 : $S \longrightarrow 0 \mid \bullet R\sharp$ $(Initialisierung)$
02 : $R \longrightarrow 0 \mid \bullet aRb$ $Vorausschau(1)$
03 : $R \longrightarrow 0 \mid \bullet A$ $Vorausschau(1)$
04 : $A \longrightarrow 0 \mid \bullet Aa$ $Vorausschau(3)$
05 : $A \longrightarrow 0 \mid \bullet a\ \}$ $Vorausschau(3)$

$Q_1 = \{$
06 : $R \longrightarrow 0 \mid a \bullet Rb$ $Eingabevergleich(2)$
07 : $A \longrightarrow 0 \mid a \bullet$ $Eingabevergleich(5)$

$08: \quad R \longrightarrow 1 \mid \bullet aRb \quad Vorausschau(6)$
$09: \quad R \longrightarrow 1 \mid \bullet A \quad Vorausschau(6)$
$10: \quad R \longrightarrow 0 \mid A \bullet \quad Rekonstruktion(7, mit\ 3)$
$11: \quad A \longrightarrow 0 \mid A \bullet a \quad Rekonstruktion(7, mit\ 4)$
$12: \quad A \longrightarrow 1 \mid \bullet a \quad Vorausschau(9)$
$13: \quad A \longrightarrow 1 \mid \bullet Aa \quad Vorausschau(9)$
$14: \quad S \longrightarrow 0 \mid R \bullet \sharp\ \} \quad Rekonstruktion(10, mit\ 1)$

$Q_2 = \{$
$15: \quad R \longrightarrow 1 \mid a \bullet Rb \quad Eingabevergleich(8)$
$16: \quad A \longrightarrow 0 \mid Aa \bullet \quad Eingabevergleich(11)$
$17: \quad A \longrightarrow 1 \mid a \bullet \quad Eingabevergleich(12)$
$18: \quad R \longrightarrow 2 \mid \bullet aRb \quad Vorausschau(15)$
$19: \quad R \longrightarrow 2 \mid \bullet A \quad Vorausschau(15)$
$20: \quad R \longrightarrow 0 \mid A \bullet \quad Rekonstruktion(16, mit\ 3)$
$21: \quad A \longrightarrow 0 \mid A \bullet a \quad Rekonstruktion(16, mit\ 4)$
$22: \quad R \longrightarrow 1 \mid A \bullet \quad Rekonstruktion(17, mit\ 9)$
$23: \quad A \longrightarrow 1 \mid A \bullet a \quad Rekonstruktion(17, mit\ 13)$
$24: \quad A \longrightarrow 2 \mid \bullet Aa \quad Vorausschau(19)$
$25: \quad A \longrightarrow 2 \mid \bullet a \quad Vorausschau(19)$
$26: \quad S \longrightarrow 0 \mid R \bullet \sharp \quad Rekonstruktion(20, mit\ 1)$
$27: \quad R \longrightarrow 0 \mid aR \bullet b\ \} \quad Rekonstruktion(22, mit\ 6)$

$Q_3 = \{$
$28: \quad R \longrightarrow 2 \mid a \bullet Rb \quad Eingabevergleich(18)$
$29: \quad A \longrightarrow 0 \mid Aa \bullet \quad Eingabevergleich(21)$
$30: \quad A \longrightarrow 1 \mid Aa \bullet \quad Eingabevergleich(23)$
$31: \quad A \longrightarrow 2 \mid a \bullet \quad Eingabevergleich(25)$
$32: \quad R \longrightarrow 3 \mid \bullet aRb \quad Vorausschau(28)$
$33: \quad R \longrightarrow 3 \mid \bullet A \quad Vorausschau(28)$
$34: \quad R \longrightarrow 0 \mid A \bullet \quad Rekonstruktion(29, mit\ 3)$
$35: \quad A \longrightarrow 0 \mid A \bullet a \quad Rekonstruktion(29, mit\ 4)$
$36: \quad R \longrightarrow 1 \mid A \bullet \quad Rekonstruktion(30, mit\ 9)$
$37: \quad A \longrightarrow 1 \mid A \bullet a \quad Rekonstruktion(30, mit\ 13)$
$38: \quad R \longrightarrow 2 \mid A \bullet \quad Rekonstruktion(30, mit\ 19)$
$39: \quad A \longrightarrow 2 \mid A \bullet a \quad Rekonstruktion(30, mit\ 24)$
$40: \quad A \longrightarrow 3 \mid \bullet Aa \quad Vorausschau(33)$
$41: \quad A \longrightarrow 3 \mid \bullet a \quad Vorausschau(33)$
$42: \quad S \longrightarrow 0 \mid R \bullet \sharp \quad Rekonstruktion(34, mit\ 1)$
$43: \quad R \longrightarrow 0 \mid aR \bullet b \quad Rekonstruktion(36, mit\ 6)$

$44: \quad R \longrightarrow 1 \mid aR \bullet b \ \} \quad Rekonstruktion(38, mit\ 15)$

$Q_4 = \{$
$45: \quad R \longrightarrow 0 \mid aRb \bullet \quad Eingabevergleich(43)$
$46: \quad R \longrightarrow 1 \mid aRb \bullet \quad Eingabevergleich(44)$
$47: \quad S \longrightarrow 0 \mid R \bullet \sharp \quad Rekonstruktion(45, mit\ 1)$
$48: \quad R \longrightarrow 0 \mid aR \bullet b \ \} \quad Rekonstruktion(46, mit\ 6)$

$Q_5 = \{$
$49: \quad S \longrightarrow 0 \mid R\sharp \bullet \ \} \quad Eingabevergleich(47)$
Somit wird die Eingabe akzeptiert.

4.8 Deterministische Kellerautomaten und deterministische kontextfreie Sprachen

Da zu jedem endlichen Automaten ein äquivalenter deterministischer Automat existiert, kann jede reguläre Sprache L durch einen deterministischen endlichen Automaten A erkannt werden. A liefert gleichzeitig einen deterministischen und backtrackingfreien Algorithmus, um zu entscheiden, ob ein Eingabewort zu L gehört. Wir gehen nun der Frage nach, unter welchen Umständen man auch für eine *kontextfreie* Sprache L einen deterministischen Kellerautomaten konstruieren kann, der eine ähnliche Funktionalität bietet. Zunächst müssen wir die Frage klären, was unter einem deterministischen Kellerautomaten zu verstehen ist. Um unnötige Einschränkungen zu vermeiden, ist es an dieser Stelle sinnvoll, das Konzept des Kellerautomaten unwesentlich zu verallgemeinern:

Definition 4.41 Sei $k > 0$ eine natürliche Zahl. Ein *Kellerautomat mit $V^{(k)}$-Übergangsrelation* (V für Vorausschau) ist ein Sechstupel $A = (Q, \Sigma, \Gamma, \Delta, s, F)$, wobei Q, Σ, Γ, s, F wie üblich definiert sind und $\Delta \subseteq (Q \times \Sigma^* \times \Gamma^*) \times (Q \times \Sigma^* \times \Gamma^*)$ eine endliche Menge von Übergängen der Form

$$((q, u, \alpha), (q', u, \beta)) \quad (q, q' \in Q, u \in \Sigma^*, \alpha, \beta \in \Gamma^*) \text{ (Typ 1)}$$
und
$$((q, au', \alpha), (q', u', \beta)) \quad (q, q' \in Q, a \in \Sigma, u' \in \Sigma^*, \alpha, \beta \in \Gamma^*) \text{ (Typ 2)}$$
wo $\mid u \mid, \mid au' \mid \leq k$

Eine *Konfiguration* von A ist ein Element von $Q \times \Sigma^* \times \Gamma^*$. A erlaubt folgende Ein-Schritt Übergänge zwischen Konfigurationen:
$(q, uv, \alpha\gamma) \Longrightarrow_A^1 (q', uv, \beta\gamma)$ falls $((q, u, \alpha), (q', u, \beta)) \in \Delta$
$(q, au'v, \alpha\gamma) \Longrightarrow_A^1 (q', u'v, \beta\gamma)$ falls $((q, au', \alpha), (q', u', \beta)) \in \Delta$.
$\Longrightarrow_A^*$ bezeichnet die reflexive transitive Hülle von $\Longrightarrow_A^1$.
$w \in L(A) \iff \exists f \in F : \ (s, w, \epsilon) \Longrightarrow_A^* (f, \epsilon, \epsilon)$

Kommentar: Die Übergangsregeln machen eine ,,Vorausschau“ von (maximal) k Elementen. Hierbei wird maximal ein Symbol konsumiert.

Offenkundig kann man jeden Kellerautomaten im Sinn von Definition 4.1 auffassen als einen Kellerautomaten mit $V^{(1)}$-Übergangsrelation. Man beachte hierbei, daß man bei den Übergängen eines Kellerautomaten mit $V^{(k)}$-Übergangsrelation nicht gezwungen ist, eine Vorausschau von k Symbolen zu betreiben, auch wenn dies erlaubt ist. Wir zeigen nun, daß das Konzept des Kellerautomaten mit $V^{(k)}$-Übergangsrelation nicht wirklich mächtiger ist als das Konzept des konventionellen Kellerautomaten.

Satz 4.42 *Zu jedem Kellerautomaten A mit $V^{(k)}$-Übergangsrelation existiert ein Kellerautomat A' im Sinn von Definition 4.1 mit $L(A) = L(A')$.*

Beweis: Sei $A = (Q, \Sigma, \Gamma, \Delta, s, F)$ Kellerautomat mit $V^{(k)}$-Übergangsrelation. Die Idee besteht darin, die bei A in Form einer Vorausschau erhaltene Information bei A' in die Zustände zu kodieren. Setze $A' := (Q', \Sigma, \Gamma, \Delta', s', F')$ mit

$$\begin{aligned}
Q' &:= \{q_u \mid q \in Q, u \in \Sigma^*, |u| \leq k\}, \\
s' &:= s_\epsilon, \\
F' &:= \{f_\epsilon \mid f \in F\}, \\
\Delta' &:= \{((q_u, v, \epsilon), (q_{uv}, \epsilon)) \mid q \in Q, u, v \in \Sigma^*, |uv| \leq k\} \\
&\quad \cup \{((q_{uv}, \epsilon, \alpha), (q'_{uv}, \beta)) \mid ((q, u, \alpha), (q', u, \beta)) \in \Delta, |uv| \leq k\} \\
&\quad \cup \{((q_{au'v}, \epsilon, \alpha), (q'_{u'v}, \beta)) \mid ((q, au', \alpha), (q', u', \beta)) \in \Delta, |au'v| \leq k\}.
\end{aligned}$$

Es gilt $L(A) = L(A')$. (Der Beweis bleibt dem Leser überlassen.)

Beispiel 4.43 Es sei A ein Automat für die Grammatik G mit Regel $S \longrightarrow aSb \mid \epsilon$ mit $V^{(1)}$-Übergangsrelation Δ. $Q = \{p, q\}$, $F = \{q\}$, $\Sigma = \{a, b\}$, $\Gamma = \{a, b, S\}$ und Δ wie folgt

1. $((p, \epsilon, \epsilon), (q, \epsilon, S))$
2. $((q, a, S), (q, a, aSb))$
3. $((q, a, a), (q, \epsilon, \epsilon))$
4. $((q, b, S), (q, b, \epsilon))$
5. $((q, b, b), (q, \epsilon, \epsilon))$

Akzeptanz von ab in A:
$(p, ab, \epsilon) \Longrightarrow^1 (q, ab, S) \Longrightarrow^1 (q, ab, aSb) \Longrightarrow^1 (q, b, Sb) \Longrightarrow^1 (q, b, b) \Longrightarrow^1 (q, \epsilon, \epsilon)$
Akzeptanz in A':
$(p_\epsilon, ab, \epsilon) \Longrightarrow^1 (p_a, b, \epsilon) \Longrightarrow^1 (q_a, b, S) \Longrightarrow^1 (q_a, b, aSb) \Longrightarrow^1 (q_\epsilon, b, Sb) \Longrightarrow^1 (q_b, \epsilon, Sb) \Longrightarrow^1 (q_b, \epsilon, b) \Longrightarrow^1 (q_\epsilon, \epsilon, \epsilon)$

Bemerkung 4.44 In manchen Büchern werden von vorneherein alle Kellerautomaten mit $V^{(k)}$-Übergangsrelation definiert, wobei k nicht notwendigerweise a priori eingeschränkt ist.

Definition 4.45 Der Kellerautomat A heißt *deterministisch* genau dann, wenn zu jeder Konfiguration von A maximal eine direkte Nachfolgerkonfiguration existiert.

Bemerkung 4.46 Durch Einführung einer Vorausschau gelingt es in manchen Fällen, Kellerautomaten deterministisch zu machen.

Nachdem klar ist, was unter einem deterministischen Kellerautomaten zu verstehen ist, stellt sich die Frage, welche kontextfreien Sprachen von einem deterministischen Kellerautomaten akzeptiert werden.

Definition 4.47 Eine kontextfreie Sprache L heißt *deterministisch* genau dann, wenn L von einem deterministischen Kellerautomaten akzeptiert wird.

Bemerkung 4.48 Man kann zeigen, daß die Klasse der *deterministischen* kontextfreien Sprachen - anders als die Klasse aller kontextfreien Sprachen - unter Komplementbildung abgeschlossen ist. Hieraus folgt sofort, daß nicht jede kontextfreie Sprache deterministisch kontextfrei ist.

Das Ziel der nachfolgenden Darstellungen ist es, zwei wichtige Klassen von deterministischen kontextfreien Sprache zu charakterisieren. Ausgangspunkt sind hierbei zwei spezielle Arten deterministischer Kellerautomaten. In Abschnitt 4.2 hatten gezeigt, wie man zu einer gegebenen kontextfreien Grammatik G einen passenden Top-Down- bzw. Bottom-Up Kellerautomaten A konstruieren kann, der $L(G)$ akzeptiert. Wir fragen nun, unter welchen Voraussetzungen an die Grammatik G der entsprechende Kellerautomat durch eine verfeinerte Konstruktion, die Vorausschau einer festen Anzahl k von Symbolen mit in Betracht zieht, deterministisch gemacht werden kann. Es stellt sich heraus, daß die Antwort davon abhängt, ob wir einen Top-Down-Kellerautomaten oder einen Bottom-Up Kellerautomaten konstruieren wollen.

4.8.1 Deterministische Top-Down Kellerautomaten und $LL(k)$-Grammatiken

Wir orientieren uns zunächst am Ziel, einen deterministischen Top-Down Kellerautomaten zu konstruieren. Am Ende des Kapitels gehen wir dann auf Bottom-Up-Kellerautomaten ein.

Hilfsbegriffe

Im Folgenden nehmen wir an, daß die kontextfreien Grammatiken stets eine Produktion $\hat{S} \longrightarrow S$ haben, die die einzige ist, in der $\hat{S}$ auftritt.

Definition 4.49 Sei $G = (N, T, \Pi, \hat{S})$ kontextfreie Grammatik, $w \in T^*$, $k \geq 0$. Das *k-Präfix von w* ist

$$k{:}w = \begin{cases} w\sharp & falls \quad |w| < k \\ u & falls \quad w = uv, \ |u| = k \end{cases}$$

Für alle $\alpha \in (N \cup T)^*$ ist

$$\begin{array}{rcl} First_k(\alpha) & := & \{u \in (T \cup \{\sharp\})^* \mid \alpha \longrightarrow_G^* w, w \in T^*, u = k{:}w\} \\ Follow_k(\alpha) & := & \{u \in (T \cup \{\sharp\})^* \mid \hat{S} \longrightarrow_G^* v\alpha w, \ \ v, w \in T^*, u = k{:}w\} \end{array}$$

Für $\Omega \subseteq (N \cup T)^*$ ist

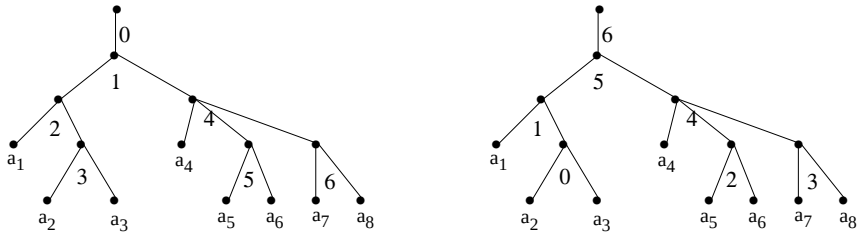


Abbildung 4.2: Parsing-Reihenfolge für $LL(k)$- (links) resp. $LR(k)$-Grammatiken.

$$\begin{array}{rcl} First_k(\Omega) & := & \bigcup_{\alpha\in\Omega} First_k(\alpha) \\ Follow_k(\Omega) & := & \bigcup_{\alpha\in\Omega} Follow_k(\alpha) \end{array}$$

Notation: $\alpha\Omega = \{\alpha\beta \mid \beta \in \Omega\}$.

Wir können nun diejenigen kontextfreien Grammatiken G beschreiben, die in einen deterministischen Top-Down Kellerautomaten mit $V^{(k)}$-Übergangsrelation übersetzt werden können.

Definition 4.50 Eine kontextfreie Grammatik $G = (N, T, \Pi, S)$ ist eine $LL(k)$-*Grammatik* genau dann, wenn für beliebige leftmost Herleitungen mit Anwendungen von Regeln $X \to \alpha$ und $X \to \alpha'$ aus Π der Form

$$\hat{S} \longrightarrow_G^* a_1 \cdots a_f X\gamma \longrightarrow_G^* \left\{ \begin{array}{ll} a_1 \cdots a_f\alpha\gamma & \longrightarrow_G^* a_1 \cdots a_f a_{f+1} \cdots a_m \\ a_1 \cdots a_f\alpha'\gamma & \longrightarrow_G^* a_1 \cdots a_f a'_{f+1} \cdots a'_n \end{array} \right.$$

(wobei $a_i, a'_i \in T$, $\gamma \in (N \cup T)^*$) stets aus $k{:}a_{f+1} \cdots a_n = k{:}a'_{f+1} \cdots a'_n$ bereits $\alpha = \alpha'$ folgt.

Die Namensgebung „$LL(k)$“ geht auf folgende Eigenschaft zurück. Bei einer Top-Down-Analyse von Links nach rechts mit einer Leftmost-Herleitung reicht eine Vorausschau von k Elementen, um eindeutig festzustellen, welche Regel angewandt wurde. Dieses Prinzip soll mit Hilfe von Abbildung 4.2 näher erläutert werden. Auf der linke Seite ist ein G-Parsebaum (nichtterminale Symbole sind weggelassen) für einen Eingabestring $a_1a_2a_3a_4a_5a_6a_7a_8$

angegeben. Die Nummern 0-6 geben die Reihenfolge der Regelanwendungen bei einer Leftmost-Herleitung mittels G an. Wir gehen davon aus, daß wir diesen Parsebaum mit Hilfe eines Topdown-Kellerautomaten ,,rekonstruieren" wollen. Dies bedeutet, daß der Kellerautomat damit beginnt, den Startzustand von G auf den Stack zu laden. Nachfolgend soll die Ersetzung nichtterminaler Symbole auf dem Keller in Art und Reihenfolge genau der Regelanwendung der Leftmost-Herleitung entsprechen. Natürlich kommen beim Kellerautomaten Eingabevergleichsschritte hinzu. Es sei nun erschwerend angenommen, daß für jedes nichtterminale Symbol A in G mehrere Regeln mit linker Seite A existieren. Im allgemeinen müssen wir dann bereits bei der Wurzel ,,raten", welche Regel zur Expansion des Startsymbols verwendet wurde. Entsprechende Rateschritte sind bei jedem weiteren inneren Knoten notwendig. Insgesamt erhalten wir ein hochgradig nichtdeterministisches Verfahren. Betrachten wir nun die Situation, wo G eine $LL(k)$-Grammatik ist, etwa für $k = 3$. Um hier festzustellen, welche Regeln an den Stellen 0,1,2 angewandt wurden, reicht es jeweils, $a_1a_2a_3$ im Rahmen einer Vorausschau anzusehen. In der Tat garantiert die $LL(3)$-Eigenschaft, daß für ein gegebenes 3:$a_1 \cdots a_8 = a_1a_2a_3$ nur eine rechte Seite α zur Ersetzung des Startsymbols in Frage kommt. Entsprechendes gilt an anderen Stellen der Analyse. Wenn wir die Regelanwendungen an den Stellen $0, 1, 2, 3$ ermittelt haben, so reicht eine Analyse von $a_4a_5a_6$, um die Regelanwendung 4 korrekt vorherzusagen. Nachfolgend werden die Regelanwendungen an den Stellen 5 und 6 durch Vorausschau ermittelt.

Wir geben nun ein Verfahren an, zu einer $LL(k)$-Grammatik einen Top-Down Kellerautomaten mit Vorausschau zu konstruieren, der genau nach der so beschriebenen Idee arbeitet.

Satz 4.51 *Zu jeder $LL(k)$-Grammatik G existiert ein deterministischer Kellerautomat A mit $V^{(k)}$-Übergangsrelation derart, daß $L(G) = L(A)$.*

Beweis: Sei $G = (N, T, \Pi, \hat{S})$. Die Zustände von A haben die Form $[X \longrightarrow \mu \bullet \nu, \Omega]$, wo $X \longrightarrow \mu\nu$ Produktionen aus Π und Ω die Menge der k-Präfixe ist, die auf X folgen können, d.h. $\Omega = Follow_k(X)$.
Konstruiere nun $A := (Q, T, \Gamma, \Delta, s, F)$ wie folgt:

1. (**Initialisierung**) Setze $q_0 := [\hat{S} \longrightarrow \bullet S, \sharp]$ und $Q := \{q_0\}$, $\Delta := \emptyset$.
2. (Zustand bearbeiten) Sei $q = [X \longrightarrow \mu \bullet \nu, \Omega]$ ein Zustand von Q, der

noch nicht behandelt wurde. In Abhängigkeit von der Gestalt von ν unterscheiden wir drei mögliche Formen der Bearbeitung:

3. (**Terminalisierung und Rekonstruktion**) Falls $\nu = \epsilon$, so füge $((q, \epsilon, *), (*, \epsilon, \epsilon))$ zu Δ. Später setze für $*$ jeden Zustand q' aus Q ein.

4. (**Eingabevergleich**) Falls $\nu = a\gamma$ für ein $a \in T, \gamma \in (N \cup T)^*$, so füge $q' := [X \longrightarrow \mu a \bullet \gamma, \Omega]$ zu Q und $((q, a, \epsilon), (q', \epsilon, \epsilon))$ zu Δ.

5. (**Vorausschau**) Falls $\nu = B\gamma$, $B \in N$, $\gamma \in (N \cup T)^*$, so sei
   $q' := [X \longrightarrow \mu B \bullet \gamma, \Omega]$ und
   $H := \{[B \longrightarrow \bullet\beta_i,\ \ First_k(\gamma\Omega)] \mid B \longrightarrow \beta_i \in \Pi\}$. Sei
   $Q := Q \cup \{q'\} \cup H$ und
   $\Delta := \Delta \cup \{((q, v_i, \epsilon), (h_i, v_i, q')) \mid h_i \in H, v_i \in First_k(\beta_i\gamma\Omega)\}$.

6. Falls alle Zustände in Q behandelt wurden, so stoppe, ansonsten gehe zu Punkt 2.

Der Automat akzeptiert $w\sharp$, falls nach Konsum von w der Zustand q_0 bei leerem Stack erreicht wird. Beim Start wird q_0 auf den Stack geladen.

Bemerkung 4.52 1. Zu jeder Regel $X \longrightarrow X_1 \cdots X_n$ der Grammatik treten im allgemeinen Zustände

$$\begin{aligned} p_0 &= [X \longrightarrow \bullet X_1 \cdots X_n, \Omega], \\ p_1 &= [X \longrightarrow X_1 \bullet X_2 \cdots X_n, \Omega], \\ &\vdots \\ p_n &= [X \longrightarrow X_1 \cdots X_n \bullet, \Omega] \end{aligned}$$

auf.
Wenn man von p_i weitergeht, muß man zu p_{i+1} (bei Akzeptanz) später zurückkehren. Dazu wird in der Vorausschau q' auf den Stapel geladen. „Weitergehen in die Tiefe“ bedeutet zunächst zu h_i zu gehen. Die Rückkehr zu q' wird später durch die Terminalisierung erzwungen, wo steht, daß der einem terminalen Zustand nachfolgende Zustand vom Stack zu entfernen ist.

2. Die Wahl von Δ in der Vorausschau erzwingt, daß man, bevor man in die Tiefe geht (Wechsel von q nach h_i), eine Vorausschau von maximal k Eingabesymbolen macht. Dafür sind die Übergänge in der Vorausschau so eingeschränkt, daß der Schritt von q nach h_i nur gemacht

wird, wenn die k folgenden Eingabesymbole konsistent sind mit der Annahme, daß tatsächlich die Regel $B \longrightarrow \beta_i$ für ein Anfangsstück des Restwortes verantwortlich ist.

3. Konsumiert wird nur, um den $\bullet$ über ein terminales Symbol hinwegzusetzen (siehe Eingabevergleich).

4. Der Algorithmus kann auf jede kontextfreie Grammatik angewandt werden. Er liefert genau dann einen deterministischen Kellerautomaten, falls G vom $LL(k)$-Typ ist.

Beispiel 4.53 G habe folgende Regeln: $Z \longrightarrow X$, $X \longrightarrow Y$, $X \longrightarrow bYa$, $Y \longrightarrow c$, $Y \longrightarrow ca$. Für $k = 3$ liefert der Algorithmus folgende Zustände und Übergänge:

1. Initialisierung:
   $q_0 = [Z \longrightarrow \bullet X, \{\sharp\}]$.

2. Vorausschau mit q_0 $(B = X, \gamma = \epsilon, \Omega = \{\sharp\})$:
   $q' = q_1$ wo
   $q_1 := [Z \longrightarrow X\bullet, \{\sharp\}]$
   $H := \{q_2, q_3\}$ wo
   $q_2 := [X \longrightarrow \bullet Y, \{\sharp\}]$,
   $q_3 := [X \longrightarrow \bullet bYa, \{\sharp\}]\}$.
   Füge zu Δ Übergänge von q_0 zu q_2 der Form
   $((q_0, c\sharp, \epsilon), (q_2, c\sharp, q_1))$, (1)
   $((q_0, ca\sharp, \epsilon), (q_2, ca\sharp, q_1))$, (2)
   und zu q_3 der Form
   $((q_0, bca, \epsilon), (q_3, bca, q_1))$. (3)

3. Terminalisierung mit q_1 ergibt Übergangsschema:
   $((q_1, \epsilon, *), (*, \epsilon, \epsilon))$. (4)

4. Vorausschau mit q_2 $(B = Y, \gamma = \epsilon, \Omega = \{\sharp\})$:
   $q' = q_4$ wo
   $q_4 := [X \longrightarrow Y\bullet, \{\sharp\}]$
   $H := \{q_5, q_6\}$ wo
   $q_5 := [Y \longrightarrow \bullet c, \{\sharp\}]$,
   $q_6 := [Y \longrightarrow \bullet ca, \{\sharp\}]\}$.
   Füge zu Δ Übergänge von q_2 zu q_5 der Form

$((q_2, c\sharp, \epsilon), (q_5, c\sharp, q_4))$ (5)
und zu q_6 der Form
$((q_2, ca\sharp, \epsilon), (q_5, ca\sharp, q_4))$. (6)

5. Eingabevergleich zu q_3 ergibt neuen Zustand
$q_7 := [X \longrightarrow b \bullet Ya, \{\sharp\}]$.
und Übergang $((q_3, b, \epsilon), (q_7, \epsilon, \epsilon))$. (7)

6. Terminalisierung mit q_4 ergibt Übergangsschema:
$((q_4, \epsilon, *), (*, \epsilon, \epsilon))$. (8)

7. Eingabevergleich zu q_5 ergibt neuen Zustand
$q_8 := [Y \longrightarrow c\bullet, \{\sharp\}]$.
und Übergang $((q_5, c, \epsilon), (q_8, \epsilon, \epsilon))$. (9)

8. (7) Eingabevergleich zu q_6 ergibt neuen Zustand
$q_9 := [Y \longrightarrow c \bullet a, \{\sharp\}]$.
und Übergang $((q_6, c, \epsilon), (q_9, \epsilon, \epsilon))$. (10)

9. Vorausschau mit q_7 ($B = Y, \gamma = a, \Omega = \{\sharp\}$):
$q' = q_{10}$ wo
$q_{10} := [X \longrightarrow bY \bullet a, \{\sharp\}]$
$H := \{q_{11}, q_{12}\}$ wo
$q_{11} := [Y \longrightarrow \bullet c, \{a\sharp\}]$,
$q_{12} := [Y \longrightarrow \bullet ca, \{a\sharp\}]$.
Füge zu Δ Übergänge von q_7 zu q_{11} bzw. q_{12} der Form
$((q_7, ca\sharp, \epsilon), (q_{11}, ca\sharp, q_{10}))$, (11)
$((q_7, caa, \epsilon), (q_{12}, caa, q_{10}))$ (12).

10. Terminalisierung mit q_8 ergibt Übergangsschema:
$((q_8, \epsilon, *), (*, \epsilon, \epsilon))$ (13).

11. Eingabevergleich zu q_9 ergibt neuen Zustand
$q_{13} := [Y \longrightarrow ca\bullet, \{\sharp\}]$.
und Übergang $((q_9, a, \epsilon), (q_{13}, \epsilon, \epsilon))$ (14).

12. Eingabevergleich zu q_{10} ergibt neuen Zustand
$q_{14} := [X \longrightarrow bYa\bullet, \{\sharp\}]$.
und Übergang $((q_{10}, a, \epsilon), (q_{14}, \epsilon, \epsilon))$ (15).

13. Eingabevergleich zu q_{11} ergibt neuen Zustand
$q_{15} := [Y \longrightarrow c\bullet, \{a\sharp\}]$.
und Übergang $((q_{11}, c, \epsilon), (q_{15}, \epsilon, \epsilon))$ (16).

14. Eingabevergleich zu q_{12} ergibt neuen Zustand
$q_{16} := [Y \longrightarrow c \bullet a, \{a\sharp\}]$.
und Übergang $((q_{12}, c, \epsilon), (q_{16}, \epsilon, \epsilon))$ (17).

15. Terminalisierung mit q_{13} ergibt Übergangsschema:
$((q_{13}, \epsilon, *), (*, \epsilon, \epsilon))$. (18).

16. Terminalisierung mit q_{14} ergibt Übergangsschema:
$((q_{14}, \epsilon, *), (*, \epsilon, \epsilon))$. (19)

17. Terminalisierung mit q_{15} ergibt Übergangsschema:
$((q_{15}, \epsilon, *), (*, \epsilon, \epsilon))$. (20)

18. Eingabevergleich zu q_{16} ergibt neuen Zustand
$q_{17} := [Y \longrightarrow ca\bullet, \{a\sharp\}]$.
und Übergang $((q_{16}, a, \epsilon), (q_{17}, \epsilon, \epsilon))$. (21)

19. Terminalisierung mit q_{17} ergibt Übergangsschema:
$((q_{17}, \epsilon, *), (*, \epsilon, \epsilon))$. (22)

Nur von den Zuständen q_0, q_2 und q_7 aus gibt es mehrere Übergänge. An den Übergängen von Zustand q_7 sieht man: wegen 3-Vorausschau ist der Automat deterministisch. Für $k = 2$ wäre ein entsprechender Automat nicht deterministisch.

Beispiel 4.54 Wir betrachten die Eingabe $bcaa\sharp$:
$(q_0, bcaa\sharp, q_0) \overset{(3)}{\Longrightarrow} (q_3, bcaa\sharp, q_1q_0) \overset{(7)}{\Longrightarrow} (q_7, caa\sharp, q_1q_0) \overset{(12)}{\Longrightarrow}$
$(q_{12}, caa\sharp, q_{10}q_1q_0) \overset{(17)}{\Longrightarrow} (q_{16}, aa\sharp, q_{10}q_1q_0) \overset{(21)}{\Longrightarrow} (q_{17}, a\sharp, q_{10}, q_1, q_0) \overset{(22)}{\Longrightarrow}$
$(q_{10}, a\sharp, q_1q_0) \overset{(15)}{\Longrightarrow} (q_{14}, \sharp, q_1q_0) \overset{(19)}{\Longrightarrow} (q_1, \sharp, q_0) \overset{(4)}{\Longrightarrow} (q_0, \sharp, \epsilon)$, also wird die Eingabe akzeptiert. Eine Vorausschau war notwendig, um ausgehend von den Zuständen $q_0 = [Z \longrightarrow \bullet X, \{\sharp\}]$ und $q_7 = [X \longrightarrow b \bullet Ya, \{\sharp\}]$ die richtigen Umformungsregeln für X und Y zu finden.

4.8.2 Deterministische Bottom-Up Kellerautomaten und LR(k) Grammatiken

Wir haben im vorausgegangenen Abschnitt gesehen, daß man für $LL(k)$-Grammatiken die Konstruktion eines Top-Down-Kellerautomaten aus G so verfeinern kann, daß wir einen *deterministischen* Kellerautomaten (mit einer

Vorausschau von k Elementen) erhalten. Die Konstruktion eines Bottom-Up-Kellerautomaten zu einer gegebenen kontextfreien Grammatik gehorcht etwas anderen Prinzipien. Wenn man ähnlich diejenigen kontextfreien Grammatiken charakterisieren möchte, zu denen wir einen deterministischen Bottom-Up-Kellerautomaten berechnen können, so kommen wir auf den nachfolgenden Begriff.

Definition 4.55 Eine kfG G heißt $LR(k)$-*Grammatik* genau dann, wenn für je zwei *Rechts*herleitungen

$$\begin{array}{rcl} S \longrightarrow_G^* \alpha Bw & \rightarrow_G^1 & \alpha\beta|w \\ S \longrightarrow_G^* \alpha' Bw' & \rightarrow_G^1 & \alpha'\beta'|w' \end{array}$$

(die Striche "|" sind nur Hilfsmarkierungen, vgl. nachfolgende Bemerkungen) mit letzter Regelanwendung $B \rightarrow \beta$ bzw. $B' \rightarrow \beta'$ aus $\alpha\beta = \alpha'\beta'$ und $First_k(w) = First_k(w')$ stets $\beta = \beta'$ sowie $B = B'$ folgt.

Informell besagt die Eigenschaft, daß es im String $\alpha\beta w$ genügt, das Teilwort $\alpha\beta$ und $First_k(w)$ zu kennen, um zweifelsfrei zu entscheiden, ob $B \rightarrow \beta$ die zuletztverwendete Regel war.

Wir werden nicht im Detail erklären, wie man zu einer $LR(k)$-Grammatik einen deterministischen Bottom-Up Kellerautomaten konstruiert. Das Akzeptanzverhalten eines solchen Kellerautomaten soll jedoch anhand Abbildung 4.2 angedeutet werden. Wir beziehen uns hierbei auf das rechte Teilbild und nehmen an, die zugrundegelegte kontextfreie Grammatik G sei eine $LR(3)$-Grammatik. Der zugehörige deterministische Kellerautomat A liest nacheinander die Symbole a_1, a_2 und a_3 auf den Stack und macht in jedem Schritt eine Vorausschau von drei Elementen. Aufgrund dieser Informationen erkennt A in den Zwischenschritten, daß weder a_1 noch a_2 oder a_1a_2 rechte Seiten einer Regelanwendung in der zugrundegelegten Rechtsableitung darstellen. Erst nachdem $a_1a_2a_3$ eingelesen sind, folgt aus der Vorausschau auf $a_4a_5a_6$, daß a_2, a_3 zu einem Nichtterminal (etwa A) zusammenzufassen sind (Regelanwendung 0). Aus der Kenntnis von a_1A (auf dem Stack liegt wie üblich das invertierte Wort Aa_1) und $a_4a_5a_6$ kann danach eindeutig die Regelanwendung 1 ermittelt werden, auf dem Stack wird Aa_1 durch das entsprechende Nichtterminal ersetzt. Im weiteren Verlauf muß nun erst $a_4a_5a_6$ eingelesen werden, bis aufgrund der Vorausschau auf a_7a_8 wieder eine Regelanwendung, in diesem Fall die Anwendung 2, erkannt wird.

Danach wird a_7a_8 eingelesen, es können nun nacheinander die Regelanwendungen $3-6$ erkannt werden. Man beachte, daß sich die zugrundegelegte Rechtsableitung durch Inversion der Reihenfolge der erkannten Regelanwendungen ergibt. Bei dieser werden nacheinander die Regelanwendungen $6,\ldots,0$ durchgeführt. Man kann nun den Sinn der Strich-markierungen in den Ableitungen in Definition 4.55 erkennen: der Balken | markiert genau die Grenze zwischen Stack und Resteingabe des Automaten. Die Vorausschau erfolgt auf der Resteingabe w bzw. w'. Im Hinblick auf den Stack besteht das Problem darin, einerseits zu sehen ob eine Reduktion erfolgen soll oder ein neues Symbol von der Eingabe auf den Stack geladen werden soll, im ersteren Fall dann die Zahl der zur Reduktion zu verwendenden Stackelemente (also β) zu bestimmen, und die korrekte linke Regelseite (Reduktionsergebnis B) zu finden.

4.8.3 Beseitigung von Linksrekursion und Grammatiken in Greibach-Normalform

Definition 4.56 Eine kontextfreie Sprache L heißt *deterministisch* genau dann, wenn sie von einem deterministischen Kellerautomaten akzeptiert wird.

Bemerkung 4.57 Es gibt viele kontextfreie Sprachen, die nicht deterministisch sind.
Beispiel: Für die Sprache L gelte
a) Es existiere für jedes $u \in T^*$ ein $v \in T^*$ mit $uv \in L$ und
b) Es sei $T^* \setminus L$ nicht kontextfrei.
Dann ist auch L nicht deterministisch kontextfrei.

Wir haben ein Verfahren kennengelernt, wie man aus einer kontextfreien Grammatik einen Kellerautomaten A konstruiert, der deterministisch ist, falls G eine $LL(k)$-Grammatik ist.
Falls A nicht deterministisch ist, kann es an der speziellen Grammatik liegen, d.h. unter Umständen existiert eine kontextfreie Grammatik G' mit $L(G) = L(G')$, bei der man einen deterministischen Kellerautomaten erhält. Eine typische Quelle von Nichtdeterminiertheit, die in der Grammatik liegt, aber nicht notwendig in der Sprache, ist die folgende:

Definition 4.58 [linksrekursiv] Eine kontextfreie Grammatik mit $A \in N$

und $A \longrightarrow_G^* A\beta$, $\beta \in (N \cup T)^+$ heißt *linksrekursiv in A*.

Lemma 4.59 *Sei G kontextfreie Grammatik und linksrekursiv in $A \in N$. Dann ist G für kein $k \in N$ eine LL(k)-Grammatik.*

Lemma 4.60 *Zu jeder linksrekursiven kontextfreien Grammatik G, wo L(G) nicht das leere Wort enthält, gibt es eine äquivalente kontextfreie Grammatik G' ohne Linksrekursionen, die algorithmisch bestimmt werden kann.*

Beweis: Es bezeichne N die Menge der nichtterminalen Symbole von G. Wir nehmen einfachheitshalber an, daß in G die rechte Seite jeder Regel entweder zumindest zwei Symbole enthält oder aber aus einer nichtleeren Folge von terminalen Symbolen besteht (vgl. Konstruktion der Chomsky-Normalform). Sei $X_1, \cdots, X_n$ eine Ordnung auf N. Falls $X_i \longrightarrow X_j\beta$ stets $i < j$ impliziert, dann ist G nicht linksrekursiv. Falls so eine Ordnung nicht existiert, führen wir für $i = 1, \ldots, |N|$ die folgenden Schritte durch (für jedes i jeweils beide Schritte, bevor der Index erhöht wird).

1. Für $j = 1, \cdots, i-1$ ersetze jede Produktion der Form

   $$X_i \longrightarrow X_j w \in \Pi$$

   durch die Menge der Produktionen

   $$\{X_i \longrightarrow \gamma_j w \mid X_j \longrightarrow \gamma_j \in \Pi\}$$

   (Danach impliziert $X_i \longrightarrow^+ X_j\delta$ stets $i \leq j$).

2. Es sei $X_i \to X_i\beta_1, \ldots, X_i \to X_i\beta_k$ eine Aufzählung aller Produktionsregeln für X_i, wo die rechte Seite mit X_i beginnt, es seien $X_i \to \gamma_1, \ldots, X_i \to \gamma_l$ die übrigen Produktionsregeln für X_i. Wir führen ein neues Hilfssymbol H ein und ersetzen diese zwei Regelmengen durch Regeln der Form $X_i \to \gamma_1 H, \ldots, X_i \to \gamma_l H$ und $H \to \beta_1 H, \ldots, H \to \beta_k H$ sowie $H \to \epsilon$.

Alle von den eingeführten Hilfssymbolen H verschiedenen Symbole werden nachfolgend „Nichthilfssymbole“ genannt. Regeln, deren linke Seite ein

Hilfssymbol ist, werden nachfolgend Hilfsregeln genannt, Regeln der Form $X_j \to \alpha$ heißen X_j-Regeln ($1 \leq j \leq |N|$).

Es ist nicht schwer zu zeigen, daß die obigen Ersetzungsschritte die erzeugte Sprache nicht verändern. Es ist weiter induktiv leicht zu verifizieren, daß nach Schritt i gilt:

1. Für $1 \leq j \leq i$: für jede X_j-Regel $X_j \to \beta$ ist β nichtleer und beginnt mit einem terminalen Symbol oder einem Symbol X_k wo $j < k$. Im letzteren Fall beginnt β mit zumindest 2 Nichthilfssymbolen.

2. Für jede Hilfsregel $H \to \beta$ ist β leer oder es beginnt mit einem terminalen Symbol oder mit zwei Nichthilfssymbolen.

Hieraus ergibt sich, daß nach Schritt $|N|$ die erhaltene Grammatik keine Linksrekursion enthält.

Beispiel 4.61 Gegeben sei eine Grammatik mit den Regeln $X_1 \longrightarrow X_2a$, $X_2 \longrightarrow X_2b$, $X_2 \longrightarrow a$.
für $i = 1$: nichts zu tun
für $i = 2$: $X_2 \longrightarrow X_2b$ wird ersetzt durch $H \longrightarrow bH$, $H \longrightarrow \epsilon$ und $X_2 \longrightarrow a$ wird ersetzt durch $X_2 \longrightarrow aH$.
Man erhält eine Grammatik G' mit den Produktionen $X_1 \longrightarrow X_2a$, $X_2 \longrightarrow aH$, $H \longrightarrow bH \mid \epsilon$.

Definition 4.62 Eine kontextfreie Grammatik G ist in *Greibach-Normalform* genau dann, wenn jede rechte Seite einer Produktionsregel mit einem terminalen Symbol beginnt.

Lemma 4.63 *Zu jeder kontextfreien Grammatik G, wo $L(G)$ nicht das leere Wort ϵ enthält, kann eine äquivalente kontextfreie Grammatik G' ein Greibach-Normalform konstruiert werden.*

Zum Beweis konstruiert man zunächst eine zu G äquivalente Grammatik G_1 ohne Linksrekursion (s.o.). Es seien $X_1, \ldots X_n$ die nichtterminalen Symbole von G_1, die so angeordnet seien, daß für jede Regel $X_i \to X_j\alpha$ von G_1 stets $i < j$ gilt. Wenn in G_1 eine Regel der Form $X_i \to X_j\alpha$ existiert,

so können wir das Vorkommen von X_j durch alle rechten Seiten β von Regeln der Form $X_j \to \beta$ ersetzen. Bei diesem Schritt erhalten wir aus der Regel $X_i \to X_j\alpha$ im allgemeinen mehrere Nachfolgerregeln. Diese Vorgehen iterieren wir. Offenkundig terminiert aufgrund der gewählten Ordnung von $X_1, \ldots, X_n$ das Verfahren. Die so erhaltene Grammatik G_2 ist offenkundig zu G_1, damit zu G, äquivalent. Jede Regel von G_2 mit nichtleerer rechter Seite beginnt mit einem terminalen Symbol. Durch eine Elimination von Regeln der Form $X_i \to \epsilon$, die nach dem Schema erfolgt, wie wir es bei der Herstellung der Chomsky-Normalform angegeben hatten, erreicht man schließlich eine zu G äquivalente Grammatik in Greibach-Normalform.

4.8.4 Zu ergänzen

Entscheidbare und unentscheidbare Probleme für kontextfreie Sprachen und Grammatiken.

Nichtkontextfreiheit natürlicher Sprachen.

Aufgaben zu Kapitel 4

Aufgabe 4.1 Hier geht es darum zu zeigen, daß man bei einem leicht modifizierten Konzept des Kellerautomaten mit einem einzigen Zustand auskommen kann. Ein *Kellerautomat mit Endmarke* ist ähnlich wie ein Standard-Kellerautomat definiert, jedoch gibt es nur einen einzigen Zustand (etwa z), außerdem hat das Stackalphabet ein besonderes Symbol ♯, das verwendet wird, um das Stackende zu markieren. Ähnlich wird das Eingabewort für die Akzeptanzentscheidung mit einem ♯ am Ende versehen. Man startet also mit der Konfiguration $(z, w\sharp, \sharp)$. Die möglichen Nachfolgekonfigurationen sind wie beim Standard-Kellerautomaten definiert, Akzeptanz erfolgt, wenn man die Konfiguration $(z, \epsilon, \sharp)$ erreicht. Zeigen Sie: zu jedem Standard-Kellerautomaten A_1 gibt es einen Kellerautomaten mit Endmarke A_2 so daß $L(A_1) = L(A_2)$.

Aufgabe 4.2 Konstruieren Sie Kellerautomaten, die die folgenden Sprachen akzeptieren:
(a) $\{a^m b^n : \ m \leq n \leq 2m\}$
(b) $\{w \in \{a, b\}^* :$ w hat genau zweimal soviele b's wie a's$\}$

Aufgabe 4.3 Ist A ein DEA, so können wir A selbst verwenden, um für ein Wort w des Eingabealphabets zu entscheiden, ob $w \in L(A)$ gilt oder nicht: die Akzeptanzprozedur ist deterministisch und hat das Eingabewort nach $|w|$ Schritten aufgebraucht – wir müssen nur schauen, ob wir in einem Finalzustand gelandet sind. Gilt auch im Falle eines beliebigen EA A, daß die übliche Akzeptanzprozedur ein Entscheidungsverfahren für „$w \in L(A)$?“ darstellt (insbesondere muß sie dazu stets terminieren)? Welche Modifikationen sind u.U. notwendig? Auf welche Schwierigkeiten stößt man, wenn man einen Kellerautomaten A verwenden will, um „$w \in L(A)$?“ zu entscheiden?

Aufgabe 4.4 Die terminalen Symbole 1,2,5,(10),(50),(1) sollen für eine 1-,2-,5-,10- bzw. 50-Pfennig Münze stehen und (1) für ein Markstück.
(a) Schreiben Sie eine kontextfreie Grammatik, die Folgen von Münzen erzeugt, die zusammen eine Mark ergeben. Bis auf mögliche Änderungen in der Reihenfolge sollen alle möglichen Folgen konstruiert werden.
(b) Konstruieren Sie einen Geld-Kellerautomaten, der genau alle Folgen von Münzen akzeptiert, die zusammen eine Mark ergeben.

Aufgabe 4.5 (a) Es sei die kf. Grammatik $G = (\{S\}, \{[,]\}, \{S \to \epsilon|SS|[S]\}, S)$ gegeben, die Klammergerüste wohlgeklammerter Ausdrücke erzeugt. Wenden Sie genau das in Beweis von 4.7 angegebene Verfahren an, um einen Kellerautomaten A zu konstruieren, der genau L(G) akzeptiert. Zeigen Sie, mit welchen Schritten das Wort [[]][] akzeptiert wird (ein Akzeptanzweg genügt).
(b) Es sei A der in Teil (a) konstruierte Kellerautomat. Wenden Sie das im Beweis zu 4.11 angegebene Verfahren an, um eine kf. Grammatik H zu finden, die genau L(A) erzeugt. (Natürlich erzeugt G die Sprache L(A), es geht hier darum, den Beweis von 4.11 besser zu verstehen.) Da der Automat A vom Startzustand gleich in den Finalzustand geht und diesen dann nicht mehr verläßt, erhält man etliche unnötige Produktionsregeln, wenn man das Verfahren wörtlich anwendet. Wenn klar ist, daß eine Produktionsregel überflüssig ist, kann sie weggelassen werden. Zeigen Sie, wie H das Wort [[]][] erzeugt, indem Sie den in (a) gefundenen Akzeptanzweg für [[]][] in A nach H „übersetzen“.

Aufgabe 4.6 (a) Konstruieren Sie eine kfG in Chomsky-Normalform, die zu der folgenden Grammatik äquivalent ist (Großbuchstaben: Nichtterminale; S=Startsymbol).

$S \to H|AB,\ A \to BAB|a,\ F \to AGB,\ H \to J,\ B \to SBS|b,\ G \to BFD,\ J \to FG,\ D \to AS,\ L \to AMBMC,\ C \to c,\ K \to ABC,\ E \to SA,\ M \to \epsilon$.
(b) Es sei G die in Teil (a) konstruierte kfG. Wenden Sie den Algorithmus von Younger, Cocke und Kasami an, um zu entscheiden, ob die Worte bbabbbc bzw. ababbcb in L(G) sind.

Aufgabe 4.7 Es sei $L = \{w \in \{a, b\}^* : \text{w enthält gleich viele a und b}\}$.
(a) Zeigen Sie, daß L nicht regulär ist.
(b) Geben Sie eine kfG in Chomsky-Normalform an mit L=L(G).
(c) Wenden Sie den Akgorithmus von Yonger, Cocke, Kasami an, um zu zeigen, daß die Worte bbbaaa bzw. ababab in L(G) sind.

Aufgabe 4.8 Gegeben sei das Akzeptanzbeispiel 4.40 für den Earley-Algorithmus. Es ist leicht zu zeigen, daß die Mehrzahl der berechneten Zustände für die Akzeptanz des Wortes aaab unnötig ist. Welche Zustände werden wirklich benötigt? Wie kann man einen Ableitungsbaum für aaab aus der Darstellung ablesen? Geben Sie drei weitere Worte an, deren Ableitbarkeit man aus dem Beispiel ersehen kann, finden Sie mit dem Beispiel auch Ableitungsbäume für diese Wörter! Bietet die Darstellung einen Überblick über alle Worte der Länge≤ 5, die von G erzeugt werden?

Aufgabe 4.9 Gegeben sei die kf. Grammatik mit den Produktionen $S \to S\sharp|aS|aSb|c$ (Kleinbuchstaben sind terminal).
Verwenden Sie den Earley-Algorithmus, um zu zeigen, daß $ab\sharp$ akzeptiert wird. Berechnen Sie alle Zustände, die der Algorithmus erzeugt.

Aufgabe 4.10 Gegeben sei die kf. Grammatik G mit den Produktionen $Z \to X,\ X \to Y|bYa,\ Y \to b|aa$ (Kleinbuchstaben sind terminal).
Stellen Sie zunächst anhand Definition 4.50 fest, für welches minimale k die Grammatik eine LL(k)-Grammatik ist. Wenden Sie dann den in Satz 4.51 angegebenen Algorithmus an, um einen deterministischen Kellerautomaten mit $V^{(k)}$-Vorausschau zu konstruieren, für den L(A)=L(G) gilt.

Aufgabe 4.11 Gegeben sei die kf. Grammatik G mit den Produktionen $Z \to X,\ X \to Xb|a$ (Kleinbuchstaben sind terminal).
Die Grammatik enthält eine Linksrekursion. Zeigen Sie, daß G für kein k eine LL(k)-Grammatik ist.

Literaturverzeichnis

[1] Rüdiger Loos: *Formale Sprachen, Theorie und Anwendungen*. Wilhelm-Schickard-Institut für Informatik, Universität Tübingen

[2] Harry R. Lewis und Christos H. Papadimitriou: *Elements of the theory of Computation*. Prentice Hall 1981

[3] Michael A. Harrison: *Introduction to Formal Language Theory*. Addison-Wesley 1978

[4] A. Saloma: *Formale Sprachen*. Springer 1978