Grundkurs Theoretische Informatik

Automatentheorie und Formale Sprachen

S. Gerber

Universität Leipzig
Institut für Informatik

Inhalt

Literatur:

Brauer, W.: Automatentheorie, Teubner, Stuttgart, 1984
Becker, W.: Walter, H.: Formale Sprachen, Vieweg, Braunschweig, 1977
Diekert, V.: Automatentheorie und Formale Sprachen, Skript, Technische Universität München, 1990
Gerber, S.: Automaten und Formale Sprachen, Skript Universität Stuttgart, 1991
Hopcroft, J.E.; Ullman, J.D.: Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie, Addison-Wesley, Bonn, 1988
Hotz, G.; Estenfeld, K.: Formale Sprachen, BI-Mannheim, 1981

1. Endliche Automaten

1.1 Deterministische und Nichtdeterministische Automaten

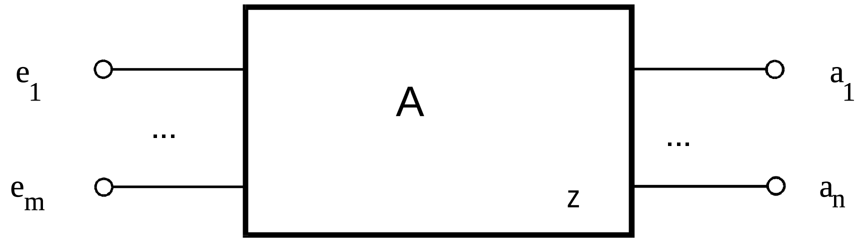
Unter einem informationsverarbeitenden System wollen wir ganz allgemein eine Einrichtung verstehen, die Nachrichten - d.h. in bestimmter Weise strukturierte Signale - aufnehmen, übertragen, speichern, umwandeln und wieder abgeben kann. Der Charakter eines solchen Systems wird u.a. durch die Mengen seiner Ein- und Ausgänge und deren Werte, seinem Zeitverhalten und der Art der Zuordnung zwischen den Ein- und Ausgangswerten bestimmt.

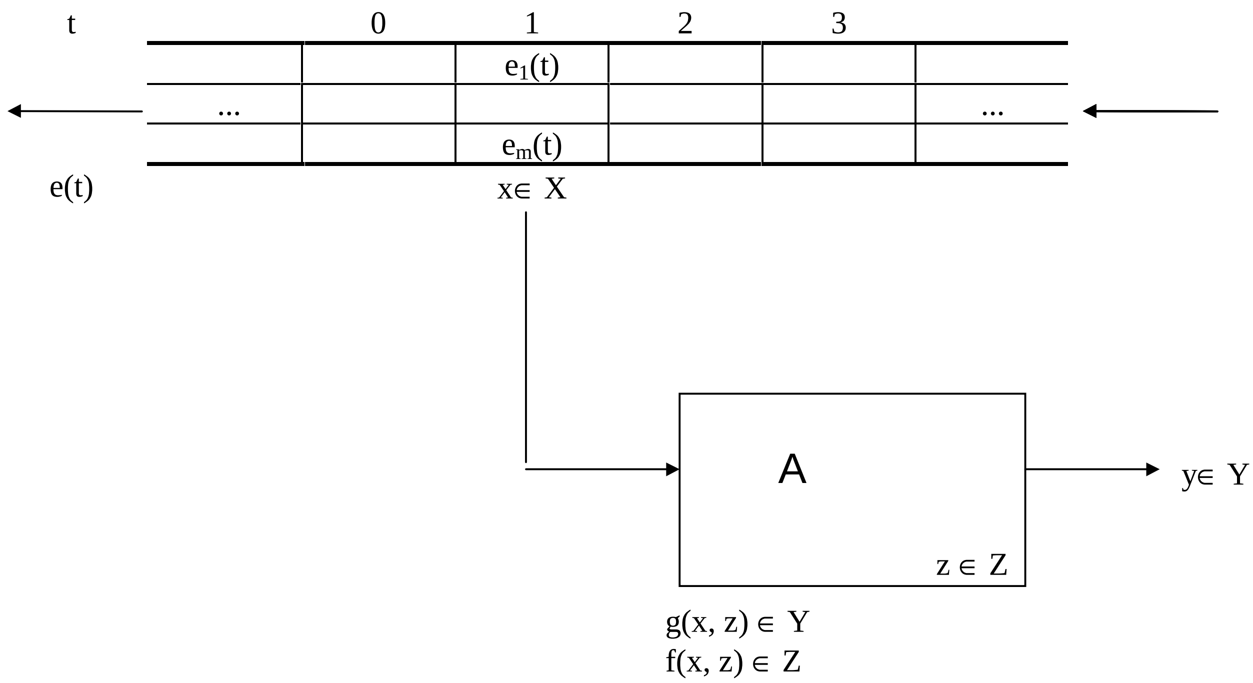
Systeme, die

- über endlich viele Ein- und Ausgänge verfügen, an denen abzählbar viele verschiedene Werte ein- bzw. ausgegeben werden können
- in einer diskreten Zeitskala arbeiten, d.h., bei denen die Zeitpunkte, die für die Beschreibung des Systemverhaltens von Bedeutung sind, eine abzählbare Menge bilden,
- determiniert sind, d.h. die Ausgangswerte sind in jedem Zeitpunkt eindeutig durch die Eingangswerte festgelegt, die zu den vorausgehenden Zeitpunkten vorgelegen hatten (Vorgeschichte)

werden als *determinierte digitale Systeme* bezeichnet. Die Eingänge $e_1, ..., e_m$ und die Ausgänge $a_1, ..., a_n$ eines determinierten digitalen Systems können als Funktionen aufgefaßt werden, die jedem Zeitpunkt (ganze Zahl) t, einen Signalwert $e_i(t)$ bzw. $a_j(t)$ aus den entsprechenden Wertmengen der Ein- bzw. Ausgänge zuordnen. Werden die verschiedenen Ein- bzw. Ausgänge zu Tupeln zusammengefaßt, dann entsteht ein System mit dem Eingang *E* und dem Ausgang *A*.
E und *A* sind Funktionen, deren Wertebereich die m-te Potenz der Eingangswertmenge und die n-te Potenz der Ausgangswertmenge bilden, und die im weiteren Eingabealphabet X bzw. Ausgabealphabet Y heißen. Die Vorgeschichte der Eingangswerte wird als Zustand z des Systems beschrieben und kann ebenfalls als Funktion *z(t)* der Zeitpunkte *t* aufgefaßt werden.
Der Begriff des abstrakten Automaten entstand im Zusammenhang mit Untersuchungen über sequentielle diskrete Schaltsysteme (Huffmann, 1954). In Arbeiten von Mealy (1955) und Moore (1956) werden abstrakte Automaten als mathematische Strukturen eingeführt mit
$E(t) = (e_1(t), ..., e_m(t)) \in X$, $A(t) = (a_1(t), ..., a_m(t)) \in Y$, $z(t) \in Z$, dessen Verhalten durch eine Überführungsfunktion $f: X \times Z \longrightarrow Z$ und eine Ergebnisfunktion $g: X \times Z \longrightarrow Y$ beschrieben wird.

Automatenmodell:





Ein endlicher Automat kann als ein gerichteter Graph dargestellt werden, indem die Knoten des Graphen den Zuständen des Automaten entsprechen. Existiert bei Eingabe x ein Übergang vom Zustand z_1 zum Zustand z_2, so wird dieser als Pfeil im gerichteten Graphen (Transitionsdiagramm) dargestellt.

Beispiel:

Dualaddierwerk: X = {00, 01, 10, 11}; Y = {0, 1}; Z = {0, 1} mit $e_i(t)$, $a_i(t) \in \{0, 1\}$
m = 2, n = 1

Berechnungsbeispiel:

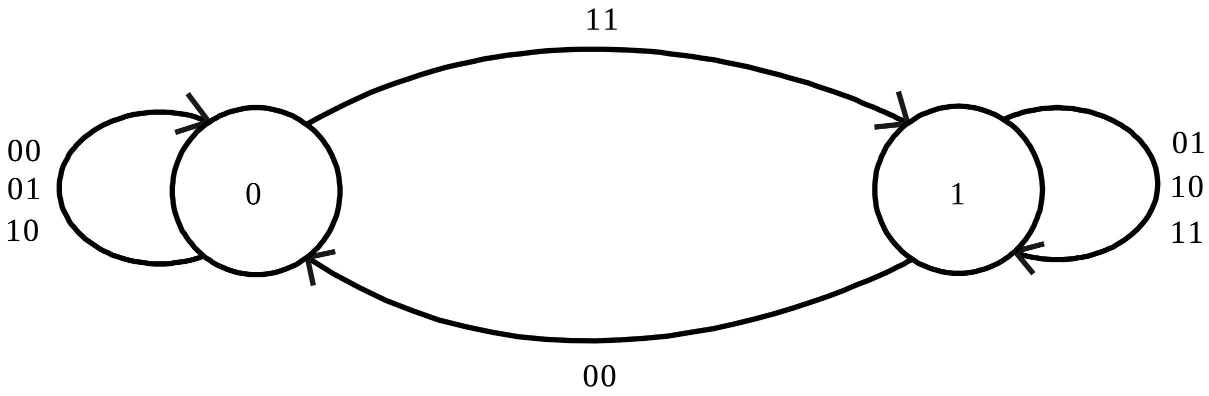
dez	dual	
07	0111	1. Summand
+ 06	+0110	2. Summand
010	01100	Übertrag
13	1101	Summe

Automatentafel:

Übertrag

x	0	1	0	1
00	0	1	0	0
01	1	0	0	1
10	1	0	0	1
11	0	1	1	1

Summe Übertrag



Transitionsdiagramm

Definition: Deterministischer Automat

Eine Struktur A = {X, Y, Z, f, g} heißt *deterministischer (endlicher) Automat*, wenn

- X, Y, Z nichtleere abzählbare (endliche) Mengen und
- f bzw. g Funktionen aus $X \times Z$ in Z bzw. Y sind.

Wir vereinbaren die Bezeichnungen:

X	Eingabemenge	x	Eingabeelement
Y	Ausgabemenge	y	Ausgabeelement
Z	Zustandsmenge	z	Zustand
f	Überführungsfunktion	f(x, z)	Folgezustand
g	Ergebnisfunktion	g(x, z)	Ergebniselement

Definiton: Vollständiger/ autonomer/initialer/ nichtdeterministischer Automat

a) A heißt *vollständiger* Automat, wenn f und g vollständig sind, d.h. für alle (x, z) aus $X \times Z$ definiert sind. (Andernfalls heißt A *partieller* Automat.)

b) A heißt *autonomer* Automat, wenn |X| = 1, d.h. X enthält nur ein Element. (Keine Abhängigkeit vom Eingang.)

c) Die Struktur (X, Y, Z, Z', f, g), wo (X, Y, Z, f, g) ein Automat und $Z' \subseteq Z$ ist (Z' Menge der Anfangszustände), heißt
- *initialer* Automat, wenn |Z'| = 1,
- *nicht-initialer* Automat, wenn Z' = Z.
- *schwach initialer* Automat, wenn $Z' \subset Z$.

d) A heißt *nicht-deterministischer* Automat, wenn f von $X \times Z$ in die Potenzmenge von Z abbildet, d.h. für alle (x, z) die Werte f(x, z) Teilmengen von Z sind.

e) Die Struktur (X, Z, Z', f) heißt *Automat ohne Ausgabe*, wenn X, Z nichtleere abzählbare Mengen sind, $Z' \subset Z$ (Menge der Anfangszustände) und f eine Funktion aus $X \times Z$ in Z (deterministischer Automat) bzw. in die Potenzmenge von Z (nichtdeterministischer Automat) ist.

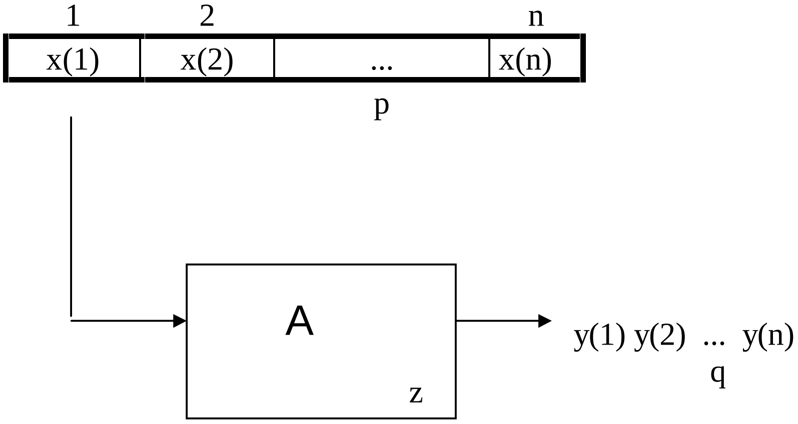
Definition: Transitionsdiagramm (Zustandsgraph)

T = (Z, K) heißt *Transitionsdiagramm (Zustandsgraph)* des endlichen Automaten ohne Ausgabe A = (X, Z, f), wenn $K = \{ (z, x, z') \mid z \in Z, x \in X, z' = f(x, z)\}$ (Kantenmenge).

(Bei nichtdeterministischen Automaten ist z' Element von f(x, z).)

Globales Verhalten:

Das globale Verhalten eines Automaten beschreibt seine Reaktion q = y(1)...y(n) (Ausgabewort) auf eine Folge (Eingabewort) p = x(1)...x(n) von Eingangswerten x(t) zu n aufeinanderfolgenden Zeitpunkten $1 \leq t \leq n$



Ausgehend vom Eingabewort p und dem Anfangszustand z = z(1) wird die Reaktion des Automaten bestimmt durch $y(t) = g(x(t), z(t))$, $z(t+1) = f(x(t), z(t))$

X* bzw. Y* sei die Menge aller Wörter (Zeichenketten) über X bzw. Y. Die Teilmengen von X* bzw. Y* heißen (formale) Sprachen über X bzw. Y. Durch l(p) wird die Länge (Anzahl der Zeichenstellen) des Wortes p bezeichnet. Das Symbol für das leere Wort sei ε (leere Zeichenkette) mit $l(\varepsilon) = 0$. Die Anzahl der mit x besetzten Stellen in p wird durch $l_X(p)$ bezeichnet.

Definition: Globale Automatenfunktion

a) *Länge e*ines Wortes $p \in X^*$: $l(\varepsilon) = 0$, $l(px) = l(p) + 1$

b) *Wortfunktion* Φ über (X, Y) mit $\Phi: X^* \longrightarrow Y^*$, $D(\Phi) \subseteq X^*$, $W(\Phi) \subseteq Y^*$
(D bezeichnet den Definitionsbereich und W den Wertebereich)

c) *Globale (deterministische)* Automatenfunktionen f' und g'

$f': X^* \times Z \longrightarrow Z$ Folgezustand
$g': X^* \times Z \longrightarrow Y^*$ Ausgabewort

mit

$f'(\varepsilon, z) = z$, $f'(px, z) = f(x, f'(p, z))$
$g'(\varepsilon, z) = \varepsilon$, $g'(px, z) = g'(p, z)\, g(x, f'(p, z))$.

(*Bemerkung*: Da f'(x, z) = f(x, z) und g'(x, z) = g(x, z) wählen wir deshalb für f' und f bzw. g' und g die gleiche Bezeichnung.)

d) *Globale nichtdeterministische* Automatenfunktion f' und g'

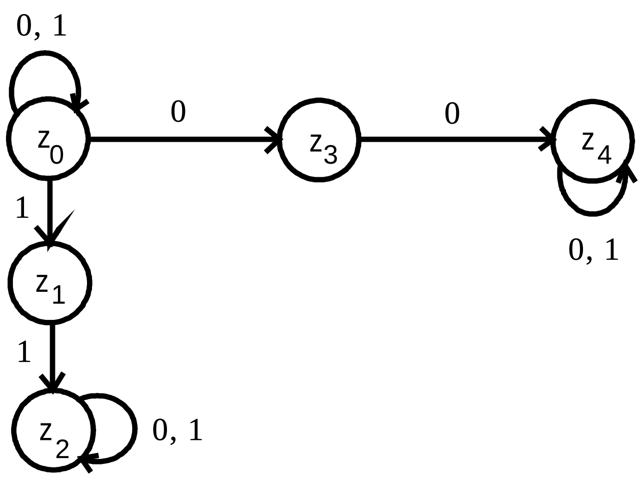
$f': X^* \times Z \longrightarrow 2^Z$ Folgezustandsmenge
$g': X^* \times Z \longrightarrow 2^{Y^*}$ Ausgabewortmenge
mit
$f'(\varepsilon, z) = \{z\}$, $f'(px, z) = \{z' \mid$ es existiert $z'' \in f'(p,z) \wedge z' \in f(x, z'')\}$,
$f'(p, Z'') = \bigcup_{z \in Z''} f'(p, z)$
$g'(\varepsilon, z) = \{\varepsilon\}$, $g'(px, z) = g'(p, z)\, g(x, f'(p,z))$ (Verkettung von Wortmengen)
$g(x, Z'') = \{g(x, z'') \mid z'' \in Z''\}$

Beispiel: Nichtdeterministischer Automat ohne Ausgabe

$X = \{0, 1\}$, $Z=\{z_0, z_1, z_2, z_3, z_4\}$

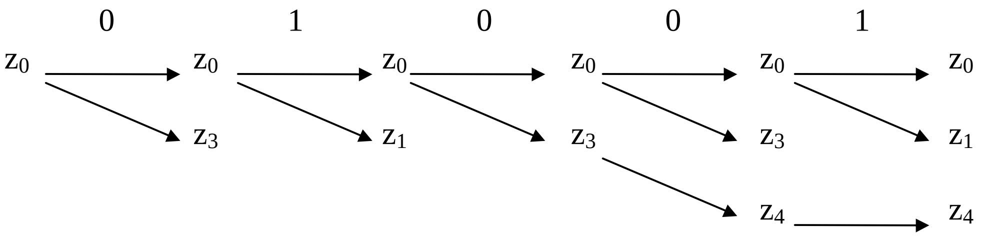
x	z_0	z_1	z_2	z_3	z_4
0	$\{z_0, z_3\}$	$\emptyset$	$\{z_2\}$	$\{z_4\}$	$\{z_4\}$
1	$\{z_0, z_1\}$	$\{z_2\}$	$\{z_2\}$	$\emptyset$	$\{z_4\}$

Automatentafel



Transitionsdiagramm

$Z' = \{z_0\}$ Anfangszustand, Eingabewort: p = 01001, Folgezustandsmenge: $\{ z_0 , z_1 ,z_4\}$
Reaktionsmöglichkeiten bei Eingabe von p:



Satz: Für alle $p, q \in X^*$ und $z \in Z$ bzw. $Z'' \subseteq Z$ gilt:
$f(pq, z) = f(q, f(p,z))$ bzw. $f(pq, Z'') = f(q, f(p, Z''))$ und
$g(pq, z) = g(p,z)\, g(q, f(p,z))$.

(Der Beweis kann induktiv über die Länge von q geführt werden und sei dem Leser als Übungsaufgabe empfohlen.)

Definiton: Erzeugte Wortfunktionen

a) Eine Wortfunktion Φ über (X, Y) heißt im Automaten A = (X, Y, Z, f, g) durch den Zustand $z \in Z$ *erzeugt*, wenn gilt:
$$g(p, z) = \Phi(p) \quad \text{für alle } p \in X^*.$$
b) Eine Wortfunktion Φ über (X, Y) heißt im Automaten A = (X, Y, Z, f, g) *erzeugbar*, wenn es einen Zustand $z \in Z$ gibt, der Φ in A erzeugt.
c) Eine Wortfunktion Φ über (X, Y) heißt *(endlich) erzeugbar*, wenn es einen Automaten A (mit endlicher Zustandsmenge) gibt, in dem Φ erzeugbar ist.

Satz: Eine Wortfunktion Φ über (X, Y) ist genau dann erzeugbar, wenn sie folgende Eigenschaften erfüllt:
a) $l(p) = l(\Phi(p))$ für alle $p \in X^*$ und
b) für jedes $p, r \in X^*$ gibt es ein $t \in Y^*$ mit $\Phi(pr) = \Phi(p)\, t$.

Bezeichnung: Wortfunktionen , die die Eigenschaften a), b) erfüllen, heißen sequentiell.
(a) Eigenschaft der Längengleichheit b) Eigenschaft der Restrospektivität)

Definition: Zustand einer Wortfunktion

Sei Φ eine sequentielle Wortfunktion über (X, Y) und $p \in X^*$. Dann heißt die Wortfunktion Φ_p *Zustand* von Φ zum Wort p, genau dann, wenn $\Phi(pq) = \Phi(p)\Phi_p(q)$ für alle $q \in X^*$.

Bezeichnung: $Z^\Phi = \{ \Phi_p \mid p \in X^*\}$ heißt die Zustandsmenge von Φ. Es gilt $\Phi = \Phi_\varepsilon \in Z^\Phi$.

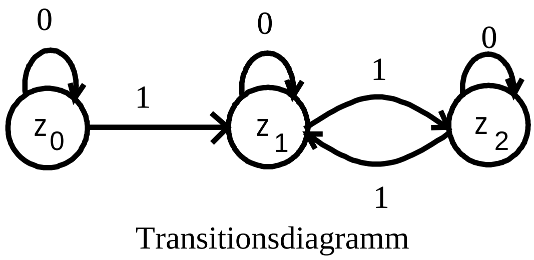
Satz: Eine sequentielle Wortfunktion Φ ist in einem *endlichen* Automaten genau dann erzeugbar, wenn die Zustandsmenge von Φ *endlich* ist.

Beispiel:

a) Die Wortfunktion Φ über (X, Y) mit X = Y = {0, 1}, $\Phi(\varepsilon) = \varepsilon$, $\Phi(px) = \Phi(p)\, y$, wobei y = 1, falls $l_1(px) > 0$ gerade, sonst 0 ist, wird in dem Automaten mit

	z_0	y	z_1	y	z_2	y
0	z_0	0	z_1	0	z_2	1
1	z_1	0	z_2	1	z_1	0

Automatentafel



Transitionsdiagramm

durch den Zustand z_0 erzeugt. Die Automatenzustände z_0 bzw. z_1 bzw. z_2 entsprechen den Zuständen $\Phi = \Phi_\varepsilon = \Phi_0$ bzw. $\Phi_1 = \Phi_{10}$ bzw. $\Phi_{11} = \Phi_{110}$ der Wortfunktion Φ.

b) Die Wortfunktion Φ über (X, Y) mit X= {x}, Y = {0, 1} und $\Phi(x^n) = y_1 ... y_n$, wo für alle $n > 0$ und $1 \leq i \leq n$: y = 1 falls i Quadratzahl und 0 sonst, ist eine sequentielle Wortfunktion aber in keinem endlichen Automaten erzeugbar, da Z^Φ *nicht* endlich ist.

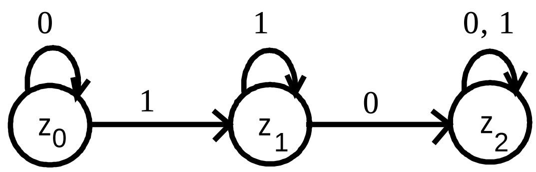
Definiton: <u>Akzeptor, akzeptierte Sprache</u>

a) Eine Struktur $A = (X, Z, f, z_0, F)$ heißt ein Akzeptor über X, wenn (X, Z, f, z_0) initialer Automat ohne Ausgabe und $F \subseteq Z$ (Menge der Finalzustände).

b) Die Menge $L(A) \subseteq X^*$ heißt die von A akzeptierte Sprache, wenn für alle $p \in X^*$ gilt:
$p \in L(A)$ gdw. $f(p, z_0) \in F$ im deterministischen Fall bzw.
$f(p, z_0) \cap F \neq \emptyset$ im nichtdeterministischen Fall.

c) Akzeptoren A_1 und A_2, die die gleichen Sprachen $L(A_1) = L(A_2)$ akzeptieren, heißen *äquivalent* (A1 ~ A2).

Beispiel: Vom Akzeptor (X, Z, f, z_0, F) mit X = {0, 1}, Z = { z_0, z_1, z_2}, F = {z_1} und der Überführungsfunktion f, dem das nachfolgende Transitionsdiagramm entspricht, wird die Sprache akzeptiert, die genau alle Wörter über X enthält, in denen alle Zeichen 0 vor allen Zeichen 1 stehen und die mindestens ein Zeichen 1 enthalten.
(*z.B.:* 1, 1...1, 01, 0...01...1)



Bemerkung:

a) Jedes zur akzeptierten Sprache gehörige Wort überführt den Akzeptor aus dem Anfangszustand z_0 in den Zustand z_1.

b) Folgt in einem Wort auf eine 1 eine 0, dann wird der Zustand z_2 erreicht, der nicht mehr verlassen werden kann ("Müllzustand").

<u>Satz</u>: Jede durch einen nichtdeterministischen endlichen Automaten akzeptierte Sprache wird durch einen deterministischen endlichen Automaten akzeptiert.

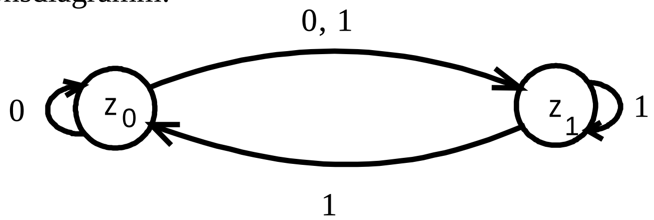
(*Beweisgedanke:* Simulation des nichtdeterministischen Automaten durch einen äquivalenten deterministischen Automaten. Die Zustände des deterministischen Automaten entsprechen dabei den Zustandsmengen des nichtdeterministischen Automaten.)

Folgerung: Deterministische endliche Automaten akzeptieren diesselbe Sprachklasse wie nichtdeterministische endliche Automaten.

Beispiel: Dem nichtdeterministischen Automaten ({0, 1}, {z_0, z_1}, f, z_0, {z_1}) mit der Automatentafel:

	z_0	z_1
0	z_0, z_1	$\emptyset$
1	z_1	z_0, z_1

bzw. dem Transitionsdiagramm:

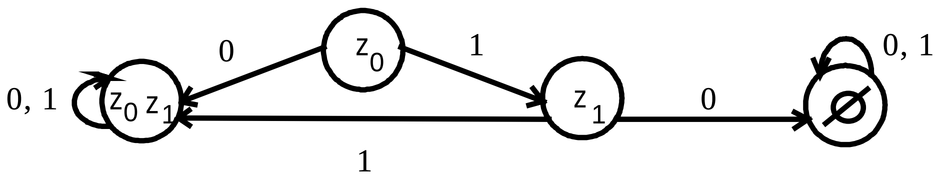


entspricht der deterministische Automat

({0,1}, {$\emptyset$, {z_0}, {z_1}, {z_0, z_1}}, f', {z_0}, {{z_1}, {z_0, z_1}}) mit der Automatentafel:

	$\emptyset$	$\{z_0\}$	$\{z_1\}$	$\{z_0, z_1\}$
0	$\emptyset$	$\{z_0, z_1\}$	$\emptyset$	$\{z_0, z_1\}$
1	$\emptyset$	$\{z_1\}$	$\{z_0, z_1\}$	$\{z_0, z_1\}$

bzw. dem Transitionsdiagramm:



Beide Automaten akzeptieren die Sprache, die genau die Wörter 1, 11p, 0q enthält, wo p, q beliebige Wörter aus {0, 1}* sind.

Definiton: Automat mit ε-Übergängen

a) A = (X, Z, f, z_0, F) heißt *Automat mit ε-Übergängen*, wenn X, Z nichtleere abzählbare Mengen sind, $z_0 \in Z$, $F \subseteq Z$ und f eine Funktion von $(X \cup \varepsilon) \times Z$ in 2^Z ist.

b) Für die globale Überführungsfunktion f' von A gilt:
$f'(\varepsilon, z) = e(z)$ und $f'(px, z) = e(f(x, f'(p, z))$ mit $z \in Z$, $x \in X$, $p \in X^*$.
e(z) enthält alle Zustände, die im Transitionsdiagramm von z durch mit ε markierte Pfeile, (u. U. über Zwischenknoten), erreichbar sind. (Es ist immer $z \in e(z)$.)

Weiter ist $f(x, Z'') = \bigcup_{z \in Z''} f(x, z)$ und $f'(p, Z'') = \bigcup_{z \in Z''} f'(p, z)$

c) Der Automat A akzeptiert die Sprache $L(A) = \{p \mid p \in X \wedge f'(p, z_0) \cap F \neq \emptyset\}$.

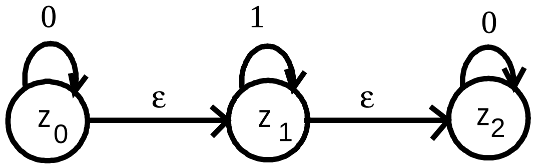
Achtung: Allgemein ist $f(x, z) \neq f'(x, z)$ und $f(\varepsilon, z) \neq f'(\varepsilon, z)$.

Beispiel: Automat A = (X, Z, f, z_0, F) mit X = {0, 1}, Z={z_0, z_1, z_2}, F= {z_2} mit

Automatentafel:

	z_0	z_1	z_2
0	z_0	∅	z_2
1	∅	z_1	∅
ε	z_1	z_2	∅

bzw. Transitionsdiagramm:

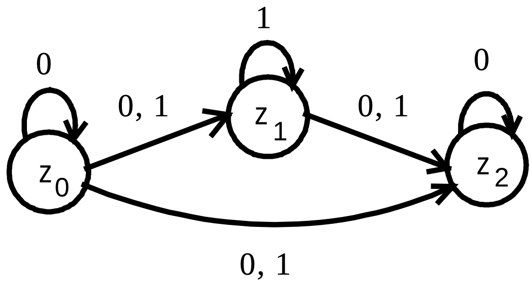


wo e(z_0) = {z_0, z_1, z_2}, e(z_1) = {z_1, z_2}, e(z_2) = {z_2}.
Das nachfolgende Diagramm bzw. Tafel zeigt einen nichtdeterministischen Automaten ohne ε-Übergänge mit den Finalzuständen z_0, z_1, z_2 , der L(A) akzeptiert.

Automatentafel:

	z_0	z_1	z_2
0	{z_0, z_1, z_2}	{z_2}	{z_2}
1	{z_1, z_2}	{z_1, z_2}	∅

Transitionsdiagramm:



Satz: Zu jedem endlichen Automaten mit ε-Übergängen gibt es einen nichtdeterministischen endlichen Automaten ohne ε-Übergänge, der dieselbe Sprache akzeptiert.

Beweisgeskizze:
Simulation des Automaten A = (X, Z, f, z_0 ,F) mit ε-Übergängen durch einen nichtdeterministischen Automaten A" = (X, Z, f", z_0 ,F") ohne ε-Übergänge, d.h. man konstruiert für den Automaten A" die Finalzustände

$$F'' = \begin{cases} F \cup \{z_0\} & \text{falls von } z_0 \text{ ein Zustand aus F nur durch } \varepsilon\text{-Übergänge erreichbar ist.} \\ F & \text{sonst} \end{cases}$$

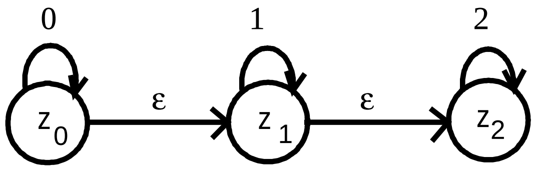
und die Überführungsfunktion f"(x, z) = f'(x, z) mit x ∈ X, z ∈ Z, wobei f' die globale Überführungsfunktion von f ist. Somit enthält A" keine ε-Transitionen.

Durch vollständige Induktion über die Länge des Eingabewortes p ist zu zeigen, daß $f''(p, z_0) = f'(p, z_0)$ gilt.

Folgerung: Endliche Automaten mit ε-Übergängen akzeptieren dieselbe Sprachklasse wie endliche Automaten ohne ε-Übergänge.

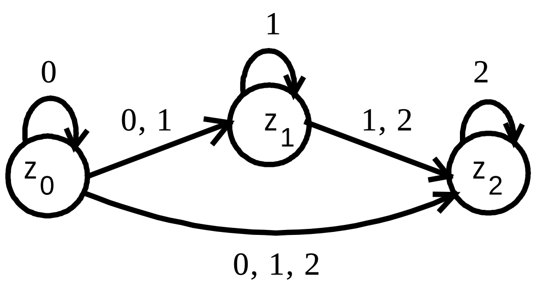
Beispiel: Der durch das Transitionsdiagramm bzw. die Automatentafel beschriebene

	z_0	z_1	z_2
0	z_0	$\emptyset$	$\emptyset$
1	$\emptyset$	z_1	$\emptyset$
2	$\emptyset$	$\emptyset$	z_2
ε	z_1	z_2	$\emptyset$



endliche Automat mit ε-Übergängen (z_0 Anfangszustand, z_2 Finalzustand) wird durch den nichtdeterministischen Automaten ohne ε-Übergänge mit dem Transitionsdiagramm bzw. der Automatentafel (z_0 Anfangszustand; z_0, z_2 Finalzustände) simuliert

	z_0	z_1	z_2
0	$\{z_0\ z_1\ z_2\}$	$\emptyset$	$\emptyset$
1	$\{z_1\ z_2\}$	$\{z_1\ z_2\}$	$\emptyset$
2	$\{z_2\}$	$\{z_2\}$	$\{z_2\}$



und erzeugt die Sprache, in der alle Zeichen 0 allen Zeichen 1 und alle Zeichen 1 allen Zeichen 2 vorangehen.

1.2 Reguläre Mengen und Reguläre Ausdrücke

Definiton: Summe, Produkt, Potenz und Iteration formaler Sprachen

Seien X eine nichtleere, endliche Zeichenmenge und L, $L_i \subseteq X^*$ formale Sprachen über X:

a) Summe (Vereinigung)

$$L_1 \cup L_2 = \{p \mid p \in L_1 \vee p \in L_2\}$$

b) Produkt (Verkettung, Concatenation)

$$L_1 \circ L_2 = \{ pq \mid p \in L_1 \wedge q \in L_2\}$$

c) Potenz

$$L^0 = \{\varepsilon\}, \ L^{n+1} = L \circ L^n \text{ für } n \geq 0$$

d) Iteration (Hülle, Stern)

$$L^* = \bigcup_{n\geq 0} L^n \ , L^+ = \bigcup_{n\geq 1} L^n \ , L^* = L^+ \cup \{\varepsilon\}$$

Beispiele: $X = \{0, 1\}$, $\{1\}^* = \{\varepsilon, 1, 11, ...\}$, $\{1\} \cup \{0\} = X$,
$\{1\}^+ \circ \{0\}^+ = \{10, 10...0, 1...10, 1...10...0\}$

Definiton: Reguläre Mengen (Sprachen) über X

a) Wenn $L = \{\varepsilon\}$ oder $L \subseteq X$, dann ist L regulär über X (Elementarsprachen)
b) Wenn L_1, L_2 regulär, dann sind $L_1 \cup L_2$, $L_1 \circ L_2$ regulär.
c) Wenn L regulär, dann ist L^* regulär.
d) L ist nur dann regulär, wenn dies nach a), b) oder c) gilt.

Bemerkung: Die Klasse der regulären Mengen (Sprachen) ist die kleinste Klasse, die alle Elementarsprachen umfaßt und abgeschlossen ist gegenüber Summe, Produkt und Iteration (Kleenesche Algebra).

Definiton: Reguläre Ausdrücke (Terme) über X

a) $\emptyset$, ε, $x \in X$ sind reguläre Ausdrücke. (Eigentlich Zeichen für $\emptyset$, ε, x.)
b) Wenn T, T_1, T_2 reguläre Ausdrücke, dann sind auch <T>, $(T_1 + T_2)$ und $(T_1 \ T_2)$ reguläre Ausdrücke.
c) T ist genau dann ein regulärer Ausdruck, wenn er nach a) oder b) gebildet ist.

Beispiel: Die Zeichenketten $(<(0 + (1\ 0))> \cdot 1)$, $<((1 + \varepsilon) \cdot 0)>$ und $<< 0 >>$ sind reguläre Ausdrücke über {0,1}.

Definiton: Interpretation Int regulärer Ausdrücke

- Int $\emptyset = \emptyset$ (leere Menge),
- Int $\varepsilon = \{\varepsilon\}$,
- Int $x = \{x\}$,
- Int $< T > =$ (Int $T)^*$,
- Int$(T_1 + T_2) =$ Int $T_1 \cup$ Int T_2,
- Int$(T_1 \cdot T_2) =$ Int $T_1 \circ$ Int T_2

Die Funktion *Int* ordnet jedem regulären Ausdruck *T* eine reguläre Menge *Int T* zu.

Klammereinsparungsregeln bei regulären Ausdrücken nach Prioritäten der Operationen:

- Iteration bindet stärker als Produkt,
- Produkt bindet stärker als Summe.

Bei geklammerten Ausdrücken in < > können Außenklammern entfallen.

Beispiele:

a) Int $(<(0 + (1 \cdot 0))> \cdot 1) =$ Int $<(0 + (1 \cdot 0))> \circ \{1\} = (\{0\} \cup (\{1\} \circ \{0\}))^* \circ \{1\}$
mit Int $<(0 + (1 \cdot 0))>$ $=$ (Int $(0 + (1 \cdot 0)))^*$

b) Int $<\emptyset> =$ (Int $\emptyset)^* = \emptyset^* = \{\varepsilon\}$, Int $<\varepsilon> =$ (Int $\varepsilon)^* = \{\varepsilon\}^* = \{\varepsilon\}$

Definiton: Äquivalenz von regulären Ausdrücken

Die regulären Ausdrücke T_1 und T_2 heißen *äquivalent*, wenn ihre Interpretationen die gleiche reguläre Menge beschreiben, d.h. $T_1 \sim T_2$ gdw. Int $T_1 =$ Int T_2

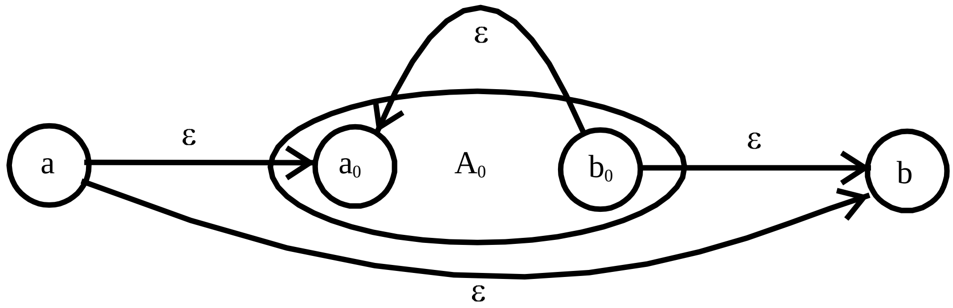
Beispiele:

a) $<\emptyset> \sim <\varepsilon> \sim \varepsilon$,
b) $(\varepsilon \cdot T) \sim (T \cdot \varepsilon) \sim T$,
c) $(\emptyset + T) \sim (T + \emptyset) \sim T$,
d) $<<T>> \sim <T>$,
e) $((T \cdot <T>) + \varepsilon) \sim <T>$,
f) $((T \cdot T_1) + (T \cdot T_2)) \sim (T \cdot (T_1 + T_2))$

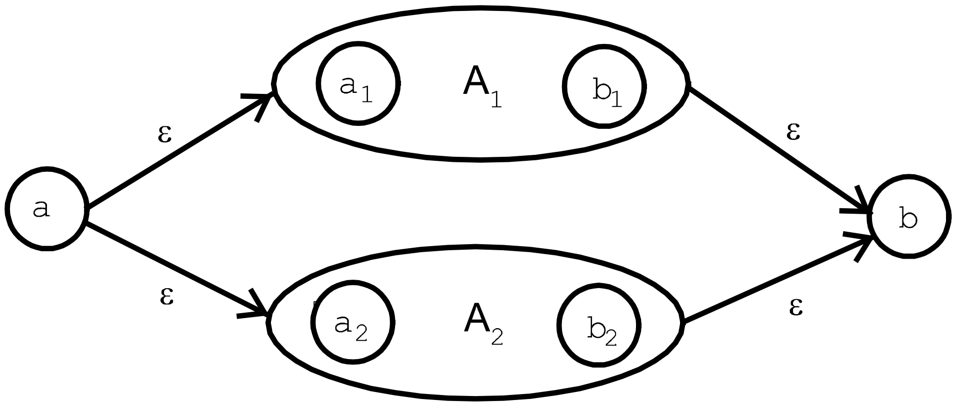
Satz: Zu jedem regulären Ausdruck T existiert ein endlicher Automat mit ε-Übergängen, der die Sprache Int T akzeptiert (Automatensynthese).

Beweis: Induktion über die Anzahl der Operatoren in T. Zusammensetzen der Automaten für die Konstituenten von T mit Hilfe von ε-Übergängen. Wenn A_0, A_1, A_2 mit den Anfangszuständen a_0, a_1, a_2 und den Finalzuständen b_0, b_1, b_2, die den Ausdrücken T_0, T_1, T_2 zugeordneten Automaten bezeichnen, dann ergeben sich für die Ausdrücke $<T_0>$, $<T_1 + T_2>$, $(T_1 \cdot T_2)$ folgende Automaten:

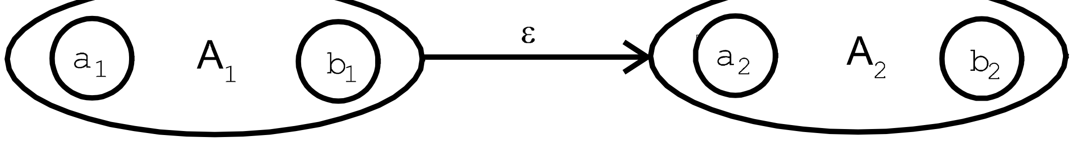
$<T_0>$



$(T_1 + T_2)$

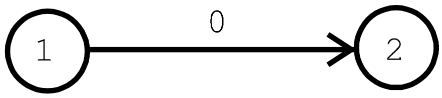


$(T_1 \cdot T_2)$

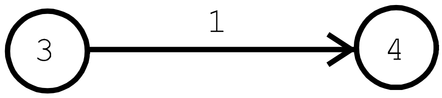


Beispiel: Konstruktion eines akzeptierenden Automaten zum Ausdruck ((0 · <1>) + 1):

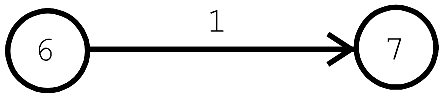
Automat für 0:



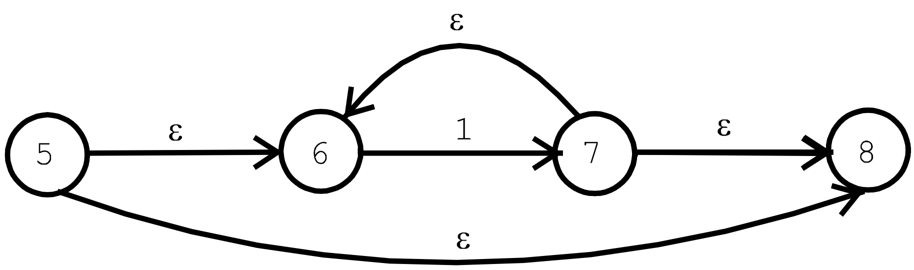
Automat für 1:



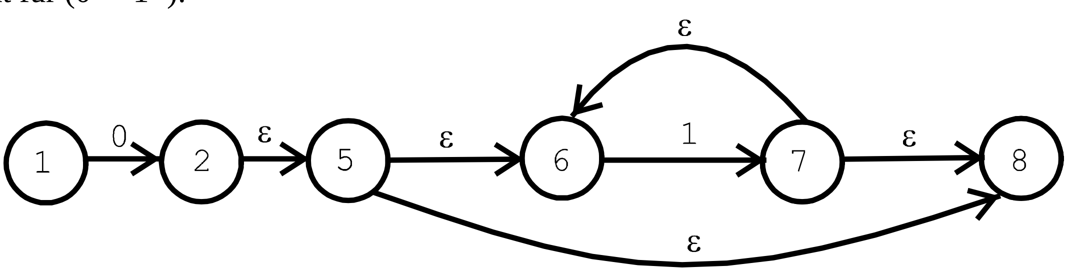
Automat für 1:



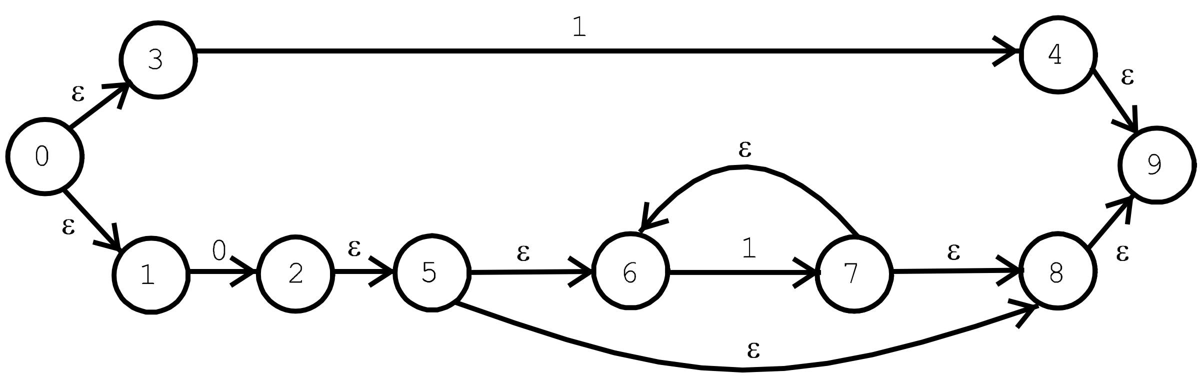
Automat für <1>:



Automat für (0 · <1>):

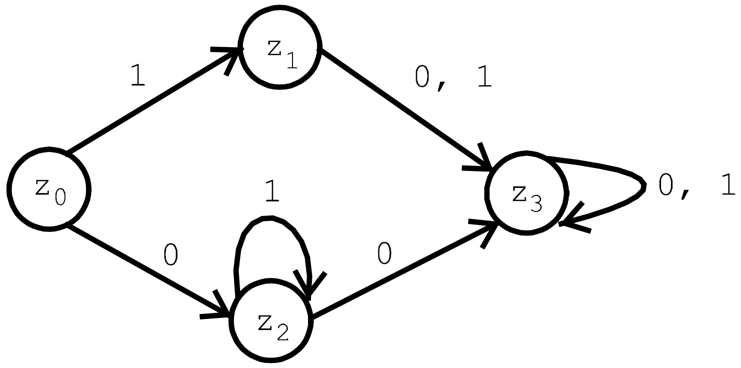


Automat für ((0 · <1>) + 1):



Folgerung: Nach den vorhergehenden Sätzen gibt es zu jedem regulären Ausdruck T einen endlichen deterministischen Automaten (ohne ε-Übergänge) A mit L(A) = Int T.

Beispiel: Die durch den Ausdruck ((0 · <1>) + 1) beschriebene Sprache (reguläre Menge) wird durch den Automaten mit dem nachfolgenden Transitionsdiagramm akzeptiert. (z_0 Anfangszustand, z_1, z_2 Finalzustände, z_3 Müllzustand).



<u>Satz:</u> Jede von einem deterministischen endlichen Automaten akzeptierte Sprache ist regulär, d.h. es existiert ein regulärer Ausdruck, der diese Sprache als Interpretation besitzt. (Automatenanalyse)

Beweisidee: Konstruktion des Ausdruckes durch Zusammensetzen aus Konstituenten, die den globalen Zustandsübergängen des gegebenen Automaten entsprechen.

Folgerung: Eine Wortmenge (Sprache) ist genau dann regulär, wenn es einen sie akzeptierenden endlichen Automaten gibt. (Reguläre Ausdrücke und endliche (nichtdeterminisitsche) Automaten (mit ε-Übergängen) beschreiben dieselbe Sprachklasse.)

<u>Analyseverfahren:</u> *(Konstruktion eines regulären Ausdruckes zu einem endlichen Automaten)*

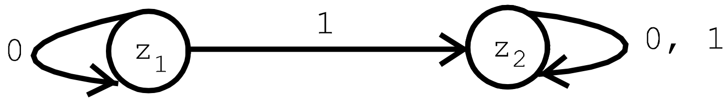
Wenn der Ausdruck T_{ij}^k die Menge aller Wörter beschreibt, die den Automaten vom Zustand z_i in den Zustand z_j überführen, und dabei höchstens die ersten k Zustände aus der Zustandsmenge Z = {z_1, ..., z_n} als Zwischenzustände auftreten, dann gilt

$$T_{ij}^k = (T_{ik}^{k-1} \cdot < T_{kk}^{k-1} > \cdot T_{kj}^{k-1}) + T_{ij}^{k-1}$$

Ausgehend von $T_{ij}^0 = \bigcup_{z_j = f(x,z_i)} x$ bzw. $(\varepsilon + \bigcup_{z_j = f(x,z_i)} x)$ falls i = j, wobei f die Überführungsfunktion des Automaten A bezeichnet, kann L(A) durch den Ausdruck $\bigcup_{z_k \in F} T_{1k}^n$ beschrieben werden.

(Dabei bezeichnet z_1 den Anfangszustand und F die Menge der Endzustände von A.)

Beispiel:
Gegeben sei der Automat A nach dem Transitionsdiagramm mit dem Anfangszustand z_1 und dem Endzustand z_2.



Die von A akzeptierte Sprache wird beschrieben durch den Ausdruck

$$T_{12}^{2} = (T_{12}^{1} \cdot < T_{22}^{1} > \cdot T_{22}^{1}) + T_{12}^{1} \text{ mit}$$
$$T_{12}^{1} = (T_{11}^{0} \cdot < T_{11}^{0} > \cdot T_{12}^{0}) + T_{12}^{0}\,,\ T_{11}^{0} = 0,\ T_{12}^{0} = 1$$
$$T_{22}^{1} = (T_{21}^{0} \cdot < T_{11}^{0} > \cdot T_{12}^{0}) + T_{22}^{0},\ T_{21}^{0} = \emptyset,\ T_{22}^{0} = (0+1)\ .$$

Damit ergibt sich $T_{22}^{1} = (\emptyset \cdot <0> \cdot 1 + (0 + 1)) \sim (0 + 1).$
$T_{12}^{1} = (0 \cdot <0> \cdot 1 + 1) \sim <0>\ 1,$
$T_{12}^{2} \sim (<0> \cdot 1 \cdot <0 + 1> \cdot (0 + 1)) + (<0> \cdot 1) \sim (<0> \cdot 1 \cdot <0 + 1>)$
und $L(A) = \{p1q \mid p \in \{0\}^* \wedge q \in \{0,1\}^*\} = \text{Int } T_{12}^{2}$.

<u>Syntheseverfahren:</u>

Konstruktion eines endlichen deterministischen Automaten als Akzeptor einer regulären Menge L über X.

1. Konstruktion über Derivatenbildung:

$D_p(L) = \{q \mid pq \in L\}$ Derivat von L bzgl. $p \in X^*$.
$D(L) = \{D_p(L) \mid p \in X^*\}$ Derivatenmenge von L

Behauptung:
a) $D_\varepsilon(L) = L$, $D_{pq}(L) = D_q(D_p(L))$.
b) Wenn L regulär, dann $D_p(L)$ regulär für alle $p \in X^*$.
c) Wenn L regulär, dann D(L) endlich.
d) Wenn L regulär, dann existiert ein endlicher Automat A= (X, D(L), f, L, F) mit $f(x, D_p(L)) = D_{px}(L)$, wobei $D_p(L) \in F$ gdw. $\varepsilon \in D_p(L)$ und $L = L(A)$.

Die Derivatenmenge von L kann sukzessive durch Derivatenbildung bzgl. x konstruiert werden. Dazu sind ausgehend vom regulären Ausdruck (Term) T, der L als Interpretation besitzt, die x-Nachfolger (x Nf T) von T mit Int(x Nf T) = D_x (Int T) zu bilden.

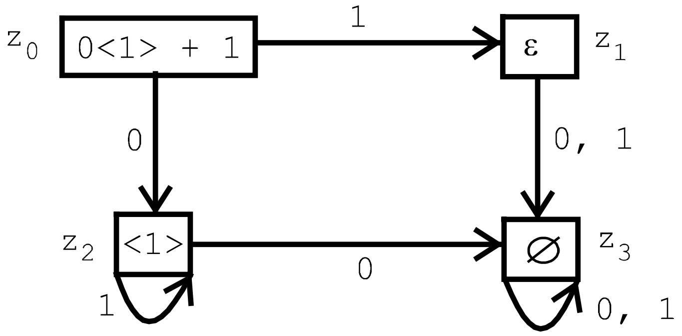
Konstruktion der x-Nachfolger von T:

a) $T = \emptyset, \varepsilon, x \in X$: x Nf T = ε falls T = x, sonst $\emptyset$.

b)

$T = \langle T_0 \rangle$:	x Nf T	$= (x \text{ Nf } T_0) \cdot T$	
$T = (T_1 + T_2)$:	x Nf T	$= (x \text{ Nf } T_1 + x \text{ Nf } T_2)$	
$T = (T_1 \cdot T_2)$:	x Nf T	$= (x \text{ Nf } (T_{11} \cdot T_2) + x \text{ Nf } (T_{12} \cdot T_2))$	falls $T_1 = (T_{11} + T_{12})$
(T_1 kein Produkt):		$= (((x \text{ Nf } T_1') \cdot T_1) + x \text{ Nf } T_2)$	falls $T_1 = \langle T_1' \rangle$
		$= T_2$	falls $T_1 = x$
		$= x \text{ Nf } T_2$	falls $T_1 = \varepsilon$
		$= \varnothing$	sonst

Beispiel: $T = 0 \cdot \langle 1 \rangle + 1$



$0 \text{ Nf } (0 \cdot \langle 1 \rangle + 1) = (\langle 1 \rangle + \varnothing) \sim \langle 1 \rangle$, $1 \text{ Nf } (0 \cdot \langle 1 \rangle + 1) = (\varnothing \cdot \langle 1 \rangle + \varepsilon) \sim \varepsilon$,
$0 \text{ Nf } \langle 1 \rangle = \varnothing \cdot \langle 1 \rangle \sim \varnothing$, $1 \text{ Nf } \langle 1 \rangle = \varepsilon \cdot \langle 1 \rangle \sim \langle 1 \rangle$, $x \text{ Nf } \varepsilon = \varnothing$, $x \text{ Nf } \varnothing = \varnothing$ für $x \in \{0,1\}$.

2. Konstruktion über Indexmengen:

Wir gehen zunächst vom Term $T = w_0 x_{i_1} w_1 \dots x_{i_n} w_n$ zu einem wiederholungsfreien Term $T' = w_0 x_{i_1}^1 w_1 \dots x_{i_n}^n w_n$ über, wo w_k Zeichenfolgen über {(,), {,}, +, e, Φ } sind und in dem jedes Alphabetzeichen an allen Stellen, in denen es im Term auftritt, unterschiedliche obere Indizes trägt. Wir bezeichnen durch $I_x(T')$ die Menge aller oberen Indizes von x in T'.

Weiter sei $E_x(T') = \{ i \,/\, \text{exist. } p \text{ mit } x^i p \in \underline{\text{Int}}\ T' \}$

die Menge aller Indizes von x, mit denen Wörter aus <u>Int</u> T' beginnen und

$N_x(I, T') = \{ j \,/\, \text{exist. } p, q \in X^*, \bar{x} \in X, i \in I \text{ mit } p\,\bar{x}^{\,i}\, x^j q \in \underline{\text{Int}}\ T' \}$ die Menge aller Indizes, mit denen x in Wörtern aus <u>Int</u> T´auf einen Index aus I folgt.

Der Automat $A = (X, Z \cup z_0, f, z_0, F)$ mit $z \subseteq I_x(T')$ für $z \in Z$, $x \in X$ und

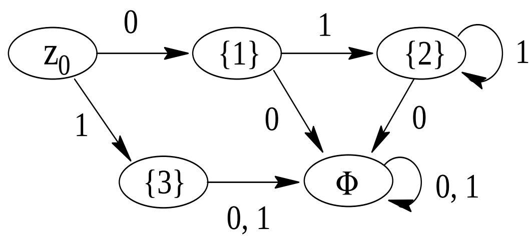
$f(x, z_0) = E_x(T') \subseteq I_x(T')$, $f(x, z) = N_x(z, T') \subseteq I_x(T')$ und

$z_0 \in F$ gdw. $e \in \underline{\text{Int}}\ T'$ bzw. $z \in F$ gdw. exist $p \in X^*$, $x \in X$, $i \in z$ mit $p x^i \in \underline{\text{Int}}\ T'$

akzeptiert $L = L(A) = \underline{\text{Int}}\ T$.

Beispiel: $T' = (0^1 < 1^2 > + 1^3)$, $E_0(T') = \{1\}$, $E_1(T') = \{3\}$, $f(0, z_0) = \{1\}$, $f(1, z_0) = \{3\}$, $f(1, \{1\}) = \{2\}$, $f(1, \{2\}) = \{2\}$, sonst $f(x, z) = \Phi$, und $F=\{\{1\}, \{2\}, \{3\}\}$
$A = (\{0, 1\}, \{z_0, 1, 2, 3, \Phi\}, f, z_0, F)$ akzeptiert die Sprache $L(A) = \underline{\text{Int}}\ T = 01^* \cup 1$.

	z_0	$\{1\}$	$\{2\}$	$\{3\}$	Φ
0	$\{1\}$	Φ	Φ	Φ	Φ
1	$\{3\}$	$\{2\}$	$\{2\}$	Φ	Φ



1.3 Eigenschaften regulärer Sprachen und endlicher Automaten

Satz Pumping-Lemma

Zu jeder regulären Menge L über X gibt es eine natürliche Zahl n_L, so daß für alle Wörter $p \in L$ mit $l(p) \geq n_L$ gilt : Es existieren Wörter $u, v, w \in X^*$ mit $l(uv) \leq n_L$ und $l(v) \geq 1$ so, daß $p = uvw$ und für alle $i \geq 0$ ist $uv^iw \in L$.

Beweis: Es existiert ein endlicher deterministischer Automat $A = (X, Z, f, z_0, F)$ mit $L(A) = L$. Wähle $n_L = |Z|$. Für ein Wort $p = xi_1 \ldots xi_m \in L$ mit $m \geq n_L$ müssen zwei der $(m + 1)$ Zustände $zi_0 = z_0$, $f(xi_1, z_0) = zi_1, \ldots, f(p, z_0) = zi_m \in F$ gleich sein. Wenn $zi_j = zi_k$, dann $k \leq n_L$, $p = uvw$ und $f(u, z_0) = zi_j$, $f(v, zi_j) = zi_k$, $f(w, zi_k) = zi_m$, d.h. $u = xi_1 \ldots xi_j$, $v = xi_{(j+1)} \ldots xi_k$, $w = xi_{(k+1)} \ldots xi_m$, $l(uv) = k \leq n_L$, $l(v) \geq 1$ und $uv^iw \in L$, denn aus $f(u, z_0) = zi_j = zi_k$, $f(w, zi_k) = zi_m$ und $f(v, zi_k) = zi_k$ folgt $f(uv^iw, z_0) = zi_m \in F$ für alle $i \geq 0$.

<u>*Bemerkung:*</u>
1) n_L kann als die Anzahl der Zustände des Automaten mit der kleinsten Zustandszahl gewählt werden, der die Sprache L akzeptiert.
2) Mit p enthält L unendlich viele Wörter der Form uv^iw .
3) Der Satz besagt nicht, daß jedes hinreichend lange Wort die Form uv^iw für ein $i > 0$ hat.
4) Wenn der Satz <u>nicht</u> gilt, dann ist L <u>nicht</u> regulär. Aus der Gültigkeit des Satzes kann aber umgekehrt nicht die Regularität geschlossen werden .

Beispiel: a) $L = \{ 0^n 1^n \mid n \geq 0 \}$ ist nicht regulär, denn sonst existiert ein $n_L \geq 0$ mit $0^{nL} 1^{nL} = uvw$, $l(uv) \leq n_L$ und $l(v) \geq 1$. Es ist demnach $u = 0^{nu}$, $v = 0^{nv}$, $w = 0^{nL-nu-nv} 1^{nL}$, $uw = 0^{nL-nu-nv} 1^{nL}$ und $n_L - n_u - n_v < n_L$ also uw nicht in L . Widerspruch zum Pumping-Lemma.

b) $L = \{ 0^{n^2} \mid n \geq 1 \}$ ist <u>nicht</u> regulär, denn sonst existiert $n_L \geq 0$ mit $0^{n_L^2} =$ uvw, $1 \leq l(v) \leq n_L$, $l(u) + l(v) + l(w) = n_L^2$, $n_L^2 = l(uvw) < l(uv^2w) = n_L^2 + l(v) \leq n_l^2 + n < (n+1)^2$ und daraus folgt uv^2w nicht in L.

Satz:

Sei A ein endlicher deterministischer Automat über X mit n Zuständen.

a) $L(A) \neq \emptyset$ genau dann, wenn ein $p \in L(A)$ mit $l(p) < n$ existiert.

b) L(A) <u>nicht endlich</u> genau dann, wenn ein $p \in L(A)$ mit $n \leq l(p) < 2n$ existiert .

Beweis: a) Wenn $L(A) \neq \emptyset$, dann wähle ein kürzestes Wort $p \in L(A)$.
Annahme: $l(p) \geq n$. Nach Pumping Lemma gilt p = uvw und $uw \in L(A)$ mit $l(uw) < l(p)$. Widerspruch zur Wahl von p. Gegenrichtung evident.

b) Sei $p \in L(A)$ mit $n \leq l(p) < 2n$. Dann gilt nach Pumping Lemma p = uvw und $uv^iw \in L(A)$ für alle $i \geq 0$, d.h., L (A) nicht endlich. Wenn L (A) nicht endlich, dann existiert ein $p \in L(A)$ mit $l(p) \geq n$.
Annahme: Es gilt immer $l(p) \geq 2n$. Für ein kürzestes dieser Wörter gilt nach Pumping Lemma p = uvw mit $1 \leq l(v) \leq n$ und $uw \in L(A)$. Also gibt es ein kürzeres Wort als p oder $l(p) < 2n$. Widerspruch zur Annahme.

Satz: <u>Komplementabgeschlossenheit</u>

Wenn $L \subseteq X^*$ regulär, dann ist auch $\overline{L} = X^* \setminus L$ regulär.

Beweis: Ist L regulär, so existiert ein Automat A mit L = L (A) und $p \in L$ gdw. $f(p, z_0) \in F$. Damit existiert ein Automat $\overline{A} = (X, Z, f, z_0, Z \setminus F)$ mit $L(\overline{A}) = X^* \setminus L$.

Satz: <u>Schnittabgeschlossenheit</u>

Wenn $L_1, L_2 \subseteq X^*$ regulär, dann ist auch $L_1 \cap L_2$ regulär.

Beweis: Zu L_i existiert ein endlicher Automat $A_i = (X, Z_i, f_i, z_{0i}, F_i)$ mit $L_i = L(A_i)$. Dann akzeptiert der Automat $A = (X, (Z_1 \times Z_2), f, z_0, F)$ mit $z_0 = (z_{01}, z_{02})$, $F = F_1 \times F_2$, $f(x, (z_1, z_2)) = (f_1(x, z_1), f_2(x, z_2))$ die Sprache $L(A) = L_1 \cap L_2$.

Satz: <u>Substitutionsabgeschlossenheit</u>

Sei $L \subseteq X^*$, $L_x \subseteq Y^*$ für jedes $x \in X$ und $h : X \to 2^{Y^*}$ mit $h(x) = L_x$,
$h(L) = \bigcup_{p \in L} h(p)$, $h(\varepsilon) = \{\varepsilon\}$, $h(px) = h(p) \circ h(x)$.

Wenn L , L_x regulär für alle $x \in X$, dann ist auch h (L) regulär.

Beweis: Für beliebige $L_i \subseteq X^*$ und $p_i \in X^*$ gilt : $h(L_1 \cup L_2) = h(L_1) \cup h(L_2)$, $h(L_1 \circ L_2) = h(L_1) \circ h(L_2)$, $h(L_0^*) = h(L_0)^*$, $h(p_1p_2) = h(p_1) \circ h(p_2)$. Aus L_x regulär folgt h (p) regulär und mit Kleeneschen Operationen auch h (L) regulär.

Satz: Homomorphieabgeschlossenheit

Es sei $h : X^* \to Y^*$ mit $h(x) \in X$ für alle $x \in X$ und $h^{-1}(L) = \{ p \mid h(p) \in L\}$. Wenn L regulär, dann ist auch $h^{-1}(L)$ regulär.

Beweis: Es existiert ein endlicher Automat $A = (X, Z, f, z_0, F)$ mit $L (A) = L$. Wir konstruieren den Automaten $\overline{A} = (X, Z, \overline{f}, z_0, F)$ mit $\overline{f}(x, z) = f(h(x), z)$. Es ist zu zeigen: $f(p, z_0) = f(h(p), z_0)$ für beliebige $p \in X^*$, d.h. $L(\overline{A}) = h^{-1}(L(A)) = h^{-1}(L)$ bzw. $p \in L(\overline{A})$ gdw. $h(p) \in L$ gdw. $p \in h^{-1}(L)$.

Beispiel: $X = \{ a, b \}$, $Y = \{ a, b, c \}$, $h_1 (a) = a$, $h_1 (b) = ba$, $h_1 (c) = a$. Dann ergibt sich $h_1^{-1}(\{ a^n\ b\ a^n \mid n \geq 1 \}) \cap a^*\ b\ c^* = \{ a^n\ b\ c^{n-1} \mid n \geq 1 \}$. Mit $h_2 (a) = 0$, $h_2 (b) = 1$, $h_2 (c) = 1$ ergibt sich weiter $h_2 (\{ a^n b c^{n-1} \mid n \geq 1 \}) = \{ 0^n 1^n \mid n \geq 1 \}$. Wenn $\{ a^n b a^n \mid n \geq 1 \}$ regulär, dann wäre auch $h_2 (h_1^{-1}(\{ a^n b a^n \mid n \geq 1 \}) \cap a^* b c^*) = \{ 0^n 1^n \mid n \geq 1 \}$ regulär.
Widerspruch zum Pumping-Lemma, also ist $\{ a^n b c^n \mid n \geq 1 \}$ nicht regulär.

Satz: Quotientenabgeschlossenheit

Sei $R \subseteq X^*$ regulär, dann ist $R / L = \{ p \mid$ existiert $q \in L$ mit $pq \in R \}$ (Quotient von R nach L) für beliebige $L \subseteq X^*$ regulär.

Beweis: Wenn R von $A = (X, Z, f, z_0, F)$ akzeptiert wird, dann wird R / L von dem Automaten $A' = (X, Z, f, z_0, F')$ mit $F' = \{ z / z \in Z$ und es existiert $q \in L$ mit $f (q, z) \in F\} \subseteq Z$ akzeptiert.
Die Konstruktion ist nicht effektiv, da $q \in L$ u.U. nicht entscheidbar ist.

Beispiel: a) $R = 0^*1^*$, $L = \{ 0^n 1^n \mid n \geq 1 \}$, $R / L = 0^*$
b) $R = 10^*1$, $L = 0^* 1^*$, $R / L = 10^*$

Definition: Äquivalenz von Automaten und Zuständen

a) Zustände $z_i \in Z_i$ der Automaten $A_i = (X, Y, Z_i, f_i, g_i)$ heißen äquivalent ($z_1 \sim z_2$), wenn $g_1 (p, z_1) = g_2 (p, z_2)$ für alle $p \in X^*$.
b) Initiale Automaten $A_i = (X, Y, Z_i, z_i, f_i, g_i)$ heißen äquivalent ($A_1 \sim A_2$), wenn $z_1 \sim z_2$.
c) Automaten $A_i = (X, Z_i, f_i, z_i, F_i)$ heißen (sprach-)äquivalent ($A_1 \sim L\ A_2$), wenn $L (A_1) = L (A_2)$.
d) Zustände $z_i \in Z_i$ der Automaten $A_i = (X, Z_i, f_i, z_{0i}, F_i)$ heißen (leistungs-)äquivalent ($z_1 \sim l\ z_2$), wenn $L (A'_1) = L (A'_2)$ mit $A'_i = (X, Z_i, f_i, z_i, F_i)$.
($L(A'_i) = L_{A_i} (z_i)$ wird als Leistung von z_i bzgl. A_i bezeichnet.)

Satz: Bestimmung äquivalenter Zustände

Für die Zustände $z_i \in Z_i$ der endliche Automaten $A_i = (X, Y, Z_i, f_i, g_i)$ gilt:
a) Aus $z_1 \sim z_2$ folgt $f_1(p, z_1) \sim f_2(p, z_2)$ für alle $p \in X^*$
b) Wenn es ein $n \geq 0$ gibt, so daß für alle $p \in X^*$ mit $l(p) = n$ gilt:
$g_1(p, z_1) = g_2(p, z_2)$ und $f_1(p, z_1) \sim f_2(p, z_2)$, dann ist $z_1 \sim z_2$.
c) Für Zustände $z_1, z_2 \in Z$ des endlichen Automaten $A=(X, Y, Z, f, g)$ mit
$|Z| = n > 0$ gilt: $z_1 \sim z_2$ genau dann, wenn $g(p, z_1) = g(p,z_2)$ für alle $p \in X^{n-1}$.

Bemerkung: Für n = 1 folgt aus a) und b): $z_1 \sim z_2$ genau dann, wenn für alle $x \in X$:
$g_1(x, z_1) = g_2(x, z_2)$ und $f_1(x, z_1) \sim f_2(x, z_2)$

Beweis: a) Aus $g_1(pq, z_1) = g_2(pq, z_2)$ für alle $pq \in X^*$ folgt: $g_1(q, f_1(p,z_1)) = g_2(q, f_2(p, z_2))$ für alle $q \in X^*$.
b) Für alle $p, q \in X^*$ mit $l(pq) \geq n$ ist
$g_1(pq, z_1) = g_1(p, z_1)\, g_1(q, f_1(p, z_1)) = g_2(p, z_2)\, g_2(q, f_2(p, z_2)) = g_2(pq, z_2)$.
c) Konstruktion der Klassen äquivalenter Zustände (Äquivalenzklassen):
Gegeben sei der endliche deterministische Automat $A = (X, Y, Z, f, g)$ mit $|Z| > 1$.
Wir bilden die Relationen $\sim_i$ über Z nach der Vorschrift:
$z_1 \sim_1 z_2$ genau dann, wenn $g(x, z_1) = g(x, z_2)$ für alle $x \in X$,
$z_1 \sim_{i+1} z_2$ genau dann, wenn $z_1 \sim_i z_2$ und $f(x, z_1) \sim_i f(x, z_2)$ für alle $x \in X$.
Damit konstuieren wir die Folge von Zerlegungen $\mathbf{Z}_i = (K^i_1, \ldots ,K^i_{ni})$ über Z mit
$K^i_j \subseteq Z$, $K^i_j \cap K^i_l = \varnothing$ für $j \neq l$ und $\bigcup_{j=1}^{n_i} K_j^i = Z$.

Behauptung: 1) ~i sind Äquivalenzrelationen über Z ($\mathbf{Z} = Z / \sim_i$ Restklassensystem von Z nach $\sim_i$.)
2) $z_1 \sim_i z_2$ genau dann, wenn $g(p, z_1) = g(p, z_2)$ für alle $p \in X^*$ der Länge $\leq$ i.
3) $n_i \leq n_{i+1}$ d.h. $\mathbf{Z}_{i+1}$ Verfeinerung von $\mathbf{Z}_i$ (n_i Index von $\sim_i$).
4) Aus $\mathbf{Z}_i = \mathbf{Z}_{i+1}$ folgt $\mathbf{Z}_i = \mathbf{Z}_{i+k}$ für alle $k \geq 1$.
5) $\mathrm{Min}\{ i \mid \mathbf{Z}_i = \mathbf{Z}_{i+1}\} \leq |Z| -1$ (Verfahren bricht ab.)
6) Aus $z_1 \sim_{n-1} z_2$ für $|Z| = n$ folgt $z_1 \sim z_2$.

Beispiel:

	z_0	z_1	z_2	z_3	z_4	z_5	z_6
0	z_4, 0	z_0, 1	z_4, 1	z_5, 1	z_0, 0	z_6, 0	z_5, 1
1	z_1, 1	z_3, 0	z_3, 0	z_2, 1	z_2, 1	z_2, 1	z, 0

entsteht die Zerlegungsfolge
$\mathbf{Z}_1 = \{ \{ z_0, z_4, z_5 \}, \{ z_1, z_2, z_6 \}, \{ z_3 \} \}$,
$\mathbf{Z}_2 = \{ \{z_0, z_4 \}, \{ z_1, z_2, z_6 \}, \{ z_3 \}, \{ z_5 \} \}$,
$\mathbf{Z}_3 = \{ \{ z_0, z_4 \}, \{ z_1, z_2 \}, \{ z_3 \}, \{ z_5 \}, \{ z_6 \} \}$,
$\mathbf{Z}_4 = \mathbf{Z}_3$,

und demnach die Zustandsäquivalenzen $z_0 \sim z_4$ und $z_1 \sim z_2$.

<u>Bemerkung:</u> a) Für den Automaten A = (X, Z, f, z_0, F) mit |Z| = n > 1 wird als Ausgangszerlegung $\mathbf{Z}_1$ = { F, Z \ F } genommen.
Dann gilt $z_1 \sim z_2$ genau dann, wenn $z_1 \sim_{n-1} z_2$.

b) Zur Konstruktion der Äquivalenzklassen wird häufig ein Ausschlußverfahren verwendet, indem diejenigen Zustandspaare bestimmt werden, die <u>nicht</u> in der Relation ~i stehen. Für den im Beispiel gegebenen Automaten entsteht damit die nachfolgende Tabelle, wobei die Zahlen die Beziehung $\not\sim_i$ für das jeweils kleinste i und * die Äquivalenz für die entsprechenden Zustandspaare angeben.

	z_0	z_1	z_2	z_3	z_4	z_5
z_1	1					
z_2	1	*				
z_3	1	1	1			
z_4	*	1	1	1		
z_5	2	1	1	1	2	
z_6	1	3	3	1	1	1

c) Für initiale endliche Automaten A = (X, Z, f, z_0, F) sei p $\sim_A$ q genau dann, wenn f (p, z_0) = f (q, z_0) .
Aus p $\sim_A$ q folgt für alle u ∈ X* : pu $\sim_A$ qu (Rechtsinvarianz).
[p] = { q | p $\sim_A$ q } sind Äquivalenzklassen bzgl. $\sim_A$. Sei L = [p] und A = (X, Z, f, z_0, F) mit z_0 = [ε], Z = { [p] | p ∈ X }, F = {[p] | p∈L}, f (x, [p]) = [px] , dann ist L = L (A).

<u>Satz:</u> <u>Entscheidbarkeit der Sprachäquivalenz</u>
Die Äquivalenzrelation ~L ist für endliche Automaten entscheidbar.

Beweis: Die von den endlichen Automaten A_i = (X, Z_i, f_i, z_i, F_i) akzeptierten Sprachen L (A_i) sind regulär und damit nach früheren Sätzen auch die Sprachen L (A_i) = X / L(A_i) , L = (L (A_1) ∩ L (A_2)) ∪ (L (A_1) ∩ L (A_2)). Es existiert demnach ein endlicher Automat A_3 mit L = L (A_3). Weiter gilt p ∈ L (A_3) genau dann, wenn p ∈ L (A_i) und p ∉ L (A_j) für i ≠ j und i,j = 1,2 , d.h., L (A_1) ≠ L (A_2).
Damit ist A_1 ~L A_2 , d.h. , L (A_1) = L (A_2) genau dann, wenn L (A_3) = ∅ . Diese Eigenschaft ist aber für endliche Automaten nach früherem Satz entscheidbar.

<u>Satz:</u> <u>Index von Äquivalenzrelationen regulärer Sprachen</u>
Für L ⊆ X* sei p R_L q genau dann, wenn für alle u ∈ X* : pu ∈ L gdw. qu ∈ L.
Die Anzahl der Äquivalenzklassen des Restklassensystems X \ R_L heißt der <u>Index</u> von R_L.
L ist genau dann regulär, wenn der Index von R_L endlich.

Beweis: a) Zu L regulär existiert ein endlicher Automat A = (X, Z, f, z_0, F) mit L(A) =L. Entsprechend der Bemerkung c) für initiale Automaten kann Z = X* | $\sim_A$ und F = { [p] | [p] ∈ L } mit [p] = Äquivalenzklasse von $\sim_A$, in der p liegt, d.h., L = { [p] | f (p, z_0) ∈ F } .

Aus [p] $\in$ X* | ~A folgt [p] $\in$ X* | ~L , denn wegen Rechtsinvarianz von ~A gilt pu $\in$ L gdw. qu $\in$ L für alle u $\in$ X* und damit p R_L q .
Also ist ~A eine Verfeinerung von R_L und der Index von R_L endlich.

b) Wenn der Index von R_L endlich ist, dann ist der oben angegebene Automat A mit z_0 = [ε] , f (x, [p]) = [px] (unabhängig von der Wahl von p aus [p]) endlich und wegen f (p, [ε]) = [p] für alle p $\in$ X* gilt L = L (A) . L ist demnach regulär.

Beispiel: Für L = 0*10* und A nach Tabelle

	z_0	z_1	z_2	z_3	z_4	z_5
0	z_1	z_0	z_4	z_4	z_4	z_5
1	z_2	z_3	z_5	z_5	z_5	z_5

mit F = { z_2, z_3, z_4 } entstehen als Klassen von $\sim_A$ die durch folgende Terme beschriebenen Wortmengen:

(00)*, (00)*0, (00)*1, (00)*01, 0*100, 0*10*1(0+1)*.

R_L besitzt die Klassen 0*, 0*10*, 0*10* 1(0+1)*.
Also wird L durch (00)* 1 + (00)*01 + 0*100* beschrieben.

Definition: Reduzierter Automat

a) Ein Automat A = (X, Y, Z, f, g) heißt reduziert , wenn für alle $z_i \in$ Z aus $z_1 \sim z_2$ folgt $z_1 = z_2$, d.h., Zustände von A sind paarweise inäquivalent.

b) Der Automat A_R heißt Reduzierter vom Automaten A, wenn A_R reduziert und zu A äquivalent ist.

Bemerkung: Nichtinitiale Automaten sind genau dann äquivalent, wenn sich ihre Zustände so aufeinander abbilden lassen, daß jeder Zustand zu seinem Bild äquivalent ist.

Satz: Existenz und Eigenschaft von Reduzierten

a) Zu jedem Automaten gibt es einen Reduzierten

b) Alle Reduzierten eines endlichen Automaten besitzen die gleiche Anzahl von Zuständen.

Beweis: a) Zu A = (X, Y, Z, f, g) konstruieren wir A_R = (X, Y, Z_R, f_R, g_R) mit Z_R = Z | ~ (Restklassensystem von Z nach ~), f_R (x, [z]) = [f (x, z)] und g_R (x, [z]) = g (x, z). (Die Definition von f_R und g_R ist unabhängig von der Auswahl der Repräsentanten aus den Äquivalenzklassen [z] von z.)
Die Automaten A und A_R sind äquivalent, da z ~ [z] für alle z $\in$ Z.
A_R ist reduziert , da aus [z] ~ [z´] folgt [z] = [z´] für alle z, z´ $\in$ Z.

b) Wenn A_R und A_R´ Reduzierte von A sind , dann gilt A_R ~ A ~ A_R´ . Wäre $|Z_R|$ < $|Z_R´|$, dann müßten verschiedene Zustände z, z´ $\in$ Z_R´ existieren, die zu einem Zustand von A_R äquivalent und damit auch untereinander äquivalent sind. A_R´ wäre demnach nicht reduziert.

Folgerung: Minimalautomat

In der Klasse aller zu einem endlichen Automaten äquivalenten Automaten gibt es bis auf (Zustands)-Isomorphie genau einen Automaten(Minimalautomat), der reduziert ist und unter allen Automaten dieser Klasse die kleinste Anzahl von Zuständen besitzt.

Beispiel: Zum Automaten aus dem vorrangegangenen Beispiel (Bestimmung äuqivalenter Zustände) entsteht der Minimalautomat mit nachfolgender Tabelle.

	z_0'	z_1'	z_3'	z_5'	z_6'
0	z_0', 0	z_0', 1	z_5', 1	z_6',0	z_5',1
1	z_1', 1	z_3', 0	z_1', 1	z_1' ,1	z_3',1

Dabei entsprechen den Zuständen die Äquivalenzklassen:

$z_0' = [\ z_0\ ] = \{\ z_0, z_4\ \}$,
$z_1' = [\ z_1\ ] = \{\ z_1, z_2\ \}$,
$z_3' = [\ z_3\ ] = \{\ z_3\ \}$,
$z_5' = [\ z_5\ ] = \{\ z_5\ \}$ und
$z_6' = [\ z_6\ ] = \{\ z_6\ \}$.

<u>*Bemerkung:*</u> a) Die Konstruktion des Minimalautomaten kann analog auch für Automaten $A = (\ X,\ Z,\ f,\ z_0,\ F\ )$ angewendet werden. Bei der Bestimmung der Äquivalenzklassen $[\ z\ ]$ wird von der Anfangszerlegung $(\ F,\ Z\backslash F\ )$ ausgegangen. Für initiale Automaten sind nur diejenigen Äquivalenzklassen als Zustände des Minimalautomaten von Interesse, die vom Anfangszustand z_0 aus erreichbare Zustände enthalten. Für endliche Automaten ist diese Eigenschaft entscheidbar und der zu A gehörige Minimalautomat $A' = (\ X,\ Z',\ f',\ z_0',\ F')$ ist bestimmt durch $Z' = \{\ [\ z\ ] \mid z$ erreichbar von $z_0\}$, $z_0' = [\ z_0\ ]$, $F' = \{\ [\ z\ ] \mid [\ z\ ] \in Z'$ und $z \in F\ \}$ und $f'(\ x, [\ z\ ]\ ) = [\ f(\ x, z\ )\ ]$, wobei die Festlegung von f' unabhängig von der Auswahl des Repräsentanten z aus [z] ist und wegen $f'(\ p,\ [\ z_0\ ]\ ) = [\ f\ (\ p,\ z_0\ )\ ]$ für alle $p \in X^*$ auch $L\ (A\ ) = L\ (\ A'\ )$.

b) Die Äquivalenz von endlichen Automaten kann durch den Vergleich der den Automaten zugeordneten Minimalautomaten überprüft werden. Die Automaten sind äquivalent, wenn es eine eineindeutige Zuordnung zwischen äquivalenten Zuständen der Minimalautomaten gibt, d.h., die Minimalautomaten zustandsisomorph sind, sich also nur in der Bezeichnung ihrer Zustände unterscheiden.

1.4 Spezielle Automaten und Anwendungen

Definition: Moore - Automat
Eine Struktur A = (X, Y, Z, f, h) heißt ein deterministischer Moore-Automat, wenn X, Y, Z nichtleere abzählbare Mengen und f bzw. h Funktionen aus $X \times Z$ bzw. Z in Z bzw. Y sind.

Bezeichnung: h heißt Markierungsfunktion und h (z) $\in$ Y Markierung von Z. Die nach bisheriger Definition eingeführten Automaten werden auch Mealy-Automaten genannt.

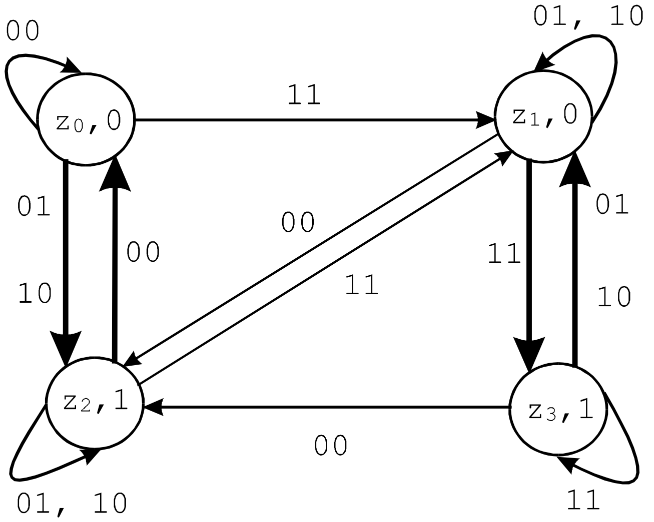
Bemerkung: Moore-Automaten können als spezielle Mealy-Automaten aufgefaßt werden, bei denen überall dort, wo derselbe Folgezustand angenommen wird, auch dieselbe Ausgabe erscheint. Die Ausgabe könnte auch in Abhängigkeit vom Vorzustand festgelegt werden. Endliche Moore-Automaten sind als Transitionsdiagramme (Moore-Diagramme) oder Tabellen darstellbar.

Definition: Moore-Diagramm
G_A = (Z, K, h) heißt Moore-Diagramm (Moore-Graph) des endlichen Moore-Automaten A = (X,Y, Z, f, h) , wenn K = { (z, x, z´) | z´= f (x, z) } mit x $\in$ X; z, z' $\in$ Z.

Beispiel: Dem Moore-Automat nach Tabelle

f	z_0	z_1	z_2	z_3
00	z_0	z_2	z_0	z_2
01	z_2	z_1	z_2	z_1
10	z_2	z_1	z_2	z_1
11	z_1	z_3	z_1	z_3
h	0	0	1	1

entspricht das Diagramm



Bei der Beschreibung des globalen Verhaltens eines Moore-Automaten ist das Ausgabewort zu bestimmen durch g (ε, z) = ε und g(px, z) = g(p, z)h (f (px, z)).
Bei der Äquivalenz $z_1 \sim z_2$ ist für Moore-Automaten neben gleichen Ergebnisfolgen g(p, z_i) für alle p ∈ X* auch die gleiche Markierung h(z_i) zu fordern.
Die anderen Begriffe lassen sich auf Moore-Automaten entsprechend übertragen.

Die Gleichwertigkeit der beiden Automatenbegriffe begründet der folgende

Satz: Äquivalenz von (Mealy-) Automaten und Moore-Automaten
Zu jedem Automaten A = (X, Y, Z, f, g) exisitiert ein äquivalenter Moore-Automat A´= (X, Y, Z´, f´, h´).

Beweis: Konstruktion des Moore-Automaten A':
Z´= Y × Z , f´(x, (y, z)) = (g (x, z), f (x, z)) und h´(y, z) = y für alle x ∈ X, y ∈ Y, z ∈ Z. Wegen g´(p, (y, z)) = g (p, z) für alle p ∈ X* , y ∈ Y, z ∈ Z, gilt (y, z) ~ z. Damit ist eine aquivalente Zuordnung der Zustände beider Automaten gegeben, also sind A und A´ äquivalent.

Bemerkung: Tatsächlich werden natürlich nur diejenigen Zustände (y, z) aus Z´ benötigt, wo z in A mit der Ausgabe y markiert ist.

Beispiel: Zu dem Automaten , der dem Dualaddierer entspricht, entsteht als äquivalenter Moore-Automat mit z_0 = (0, 0) , z_1 = (0, 1) , z_2 = (1, 0) , z_3 = (1, 1) der Automat im vorhergehenden Beispiel.

In Erweiterung der Zugriffsmöglichkeit auf das Eingabeband führen wir zweiseitige Automaten ein nach

Definition: Zweiseitiger endlicher Automat

Eine Struktur A = (X, Z, f, z_0, F) heißt zweiseitiger deterministischer endlicher Automat ,wenn X, Z nichtleere disjunkte endliche Mengen , z_0 ∈ Z (Anfangszustand), F ⊆ Z (Endzustandsmenge) und f eine Funktion (Überführungsfunktion) von X × Z in Z × { R, L } ist .

Bemerkung: Die Erweiterung der Funktion f mit Hilfe der Menge { R, L } beschreibt die Fähigkeit des Automaten das Eingabeband nach rechts (vorwärts) und nach links (rückwärts) zu lesen. f (x, z) = (z´, R) bzw. (z´, L) bedeutet, daß der Automat im Zustand z das Eingabeelement x liest , danach in den Zustand z´ übergeht und auf das nächste bzw. vorhergehende Element des Eingabebandes eingestellt wird. Das globale Verhalten eines zweiseitigen Automaten kann durch Wörter (Konfigurationen) pzq aus X* Z X* und einen Ableitungsbegriff A ├ über diesen Wörtern beschrieben werden.
Die Konfiguration pzq besagt, daß auf dem Eingabeband das Wort pq ∈ X* steht, und der Automat im Zustand z das erste Zeichen von q liest. Wenn q = ε , dann hat der Automat das rechte Ende des Eingabebandes erreicht.

Definition: Globales Verhalten zweiseitiger Automaten

Sei $A = (X, Z, f, z_0, F)$ ein zweiseitiger Automat, $p, p', q, q' \in X^*$, $x \in X$ und $z, z' \in Z$.

a) Aus dem Wort $pzxq$ heißt das Wort $p'z'q'$ direkt ableitbar (in Zeichen: $A \vdash$) genau dann, wenn $p' = px$, $q = q'$, $f(x, z) = (z', R)$ oder es gibt ein $x' \in X$ mit $p = p'x'$, $q' = x'xq$, $f(x, z) = (z', L)$.

b) $L(A) = \{ p \mid z_0 p \; A \vdash^* \; pz, \; z \in F, \; p \in X^* \}$ heißt die von A akzeptierte Sprache.

Bemerkung: Das linke Ende des Eingabebandes kann nicht überlesen werden. Für das Wort $pz\varepsilon$ ist keine Ableitung definiert. $A \vdash^*$ bezeichnet die transitive Hülle von $A \vdash$. Ein Wort p wird von A genau dann akzeptiert, wenn A im Zustand z_0 das Anfangszeichen von p liest und nach vollständiger Eingabe von p (rechtes Ende des Eingabebandes) einen Endzustand aus F erreicht .

Beispiel: Gegeben sei der zweiseitige Automat
$A = (\{0, 1\}, \{z_0, z_1, z_2\}, f, z_0, \{z_0, z_1, z_2\})$ mit der Tabelle

f	z_0	z_1	z_2
0	(z_0, R)	(z_1, R)	(z_0, R)
1	(z_1, R)	(z_2, L)	(z_2, L)

Im Zustand z_0 bewegt sich A auf dem Eingabeband solange nach rechts bis eine 1 gelesen wird. Danach geht er in den Zustand z_1 und liest das Eingabeband weiter nach rechts, bis eine zweite 1 auftritt, geht in den Zustand z_2 und führt die Operation L aus . Er bleibt in diesem Zustand, bis nach links die erste 0 gelesen wird. Dann geht er in den Zustand z_0 und beginnt an dieser Stelle des Eingabebandes erneut zu lesen. Für $p = 101001$ entsteht die Konfigurationsfolge $z_0101001$, $1z_101001$, $10z_11001$, $1z_201001$, $10z_01001$, $101z_1001$, $1010z_101$, $10100z_11$, $1010z_201$, $10100z_01$, $101001z_1$. Also ist 101001 ein Wort der von akzeptierten Sprache $L(A)$. Diese Konfigurationsfolge läßt sich auf dem Eingabeband wie folgt darstellen.

1	0	1	0	0	1	
$z_0 \rightarrow$	$z_1 \rightarrow$	z_1				
		z_2				
		$z_0 \rightarrow$	$z_1 \rightarrow$	$z_1 \rightarrow$	z_1	
					z_2	
					$z_0 \rightarrow$	z_1

Bemerkung: Folgen von Zuständen, die beim Überschreiten einer Stelle im Eingabewort angenommen werden, sollen Kreuzungsfolgen heißen. Im vorhergehenden Beispiel sind die Folgen z_0 ; z_1 und z_1, z_2, z_0 solche Kreuzungsfolgen.

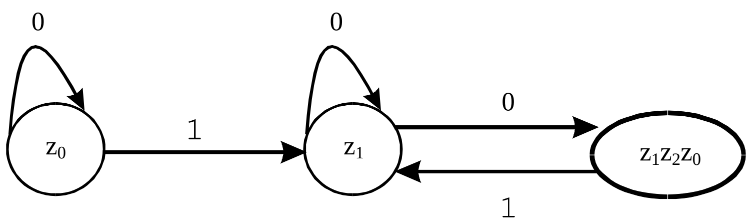
Für Kreuzungsfolgen gelten folgende Eigenschaften:

a) Wenn p von A akzeptiert wird, kann in keiner Kreuzungsfolge in der Ableitung für p ein Zustand mehrfach mit derselben Durchlaufrichtung auftreten, da sonst Zyklen entstehen und das Ende von p nicht erreicht wird. Bei n Zuständen können demnach nur Kreuzungsfolgen bis zur Länge 2n entstehen.

b) Das erste Überschreiten einer Wortstelle muß nach rechts erfolgen. Weitere Überschreitungen geschehen in alternierenden Richtungen. Beim Annehmen des i-ten Zustandes einer Kreuzungsfolge wird eine Wortstelle nach rechts bzw. nach links überschritten, wenn i ungerade bzw. gerade ist. Wenn p akzeptiert wird, haben alle in der Ableitung für p auftretenden Kreuzungsfolgen ungerade Länge.

Satz: Äquivalenz ein- und zweiseitiger endlicher Automaten
Zu jedem zweiseitigen endlichen Automaten A gibt es einen äquivalenten einseitigen endlichen Automaten A´ ,d.h. L (A) ist regulär und es gilt L(A) = L(A´) .

Beispiel: Zu dem im vorherigen Beispiel angegebenen zweiseitigen Automaten entsteht der nichtdeterministische Automat mit dem Transitionsdiagramm und den Endzuständen z_0, z_1, wenn nur die von z_0 aus erreichbaren Kreuzungsfolgen als Zustände berücksichtigt werden. L (A) = 0* + (0*1) (0 + 01)* = L (A´) = Menge aller Wörter, ohne 11 als Teilfolge.



Endliche Automaten und reguläre Mengen bzw. Ausdrücke werden bei vielen praktischen Problemen als Beschreibungs- oder Darstellungsmittel eingesetzt. Bei der lexikalischen Analyse von Programmen einer Programmiersprache z.B. können Automaten als Akzeptoren der durch reguläre Ausdrücke beschriebenen Nichtterminale eingesetzt werden. Die Verfahren zur Automatensynthese sind dabei in Generatoren als Compilermodule implementiert. In ähnlicher Weise werden Automaten in Texteditoren zur Erkennung von Zeichenketten verwendet.
Endliche Automaten dienen auch als Beschreibungsmittel für Schaltwerke. Hier wird die bei geeigneter Kodierung der Automatenalphabete bestehenden Beziehungen zu Booleschen Algebren ausgenutzt. Dazu führen wir ein

Definition: Binärkodierung

a) Jede injektive Abbildung Φ_M von einer endlichen Menge M in die Menge $\{0, 1\}^{l(M)}$ mit $l(M) \geq \log_2 |M|$ heißt eine Binärkodierung von M.

Bemerkung:
- $\Phi_M(m) \in \{0, 1\}^{l(M)}$ wird als Kodewort vom $m \in M$ bezeichnet.
- Die Länge der Kodewörter ist abhängig von der Elementeanzahl von M.

- Zur Kodierung einer n-elementigen Menge M sind Kodewörter der Länge l = kleinste ganze Zahl größer-gleich $\log_2 n$ erforderlich. Dabei sind $\prod_{k=0}^{n-1} (2^l - k)$ verschiedene Kodierungen möglich.

b) Die Abbildung $\Phi = (\Phi_X, \Phi_Y, \Phi_Z)$ heißt <u>Binärkodierung</u> des Automaten A = (X, Y, Z, f, g), wenn Φ_X bzw. Φ_Y bzw. Φ_Z Binärkodierungen der Mengen X bzw. Y bzw. Z sind.

Mit Hilfe einer Binärkodierung Φ können die Funktionen f und g des Automaten A durch l(Y) + l(Z) Boolesche Funktionen f_k ($1 \le k \le l(Z)$) und g_j ($1 \le j \le l(Y)$) wie folgt beschrieben werden :

$$f_k (\Phi_X(x), \Phi_Z(z)) = \Phi_Z (f (x, z))_k$$
$$g_j (\Phi_X(x), \Phi_Z(z)) = \Phi_Y (g (x, z))_j ,$$

wo $\Phi_Z(f (x, z))_k$ bzw. $\Phi_Y(g (x, z))_j$ die k-te bzw. j -te Stelle im Kodewort von f (x, z) bzw. g (x, z) bezeichnet. Die Binärkodierung Φ induziert einen Isomorphismus zwischen dem Automaten A und der Booleschen Struktur ({ 0, 1 }, $f_1, \ldots , f_{l(Z)}, g_1, \ldots , g_{l(Y)}$) .

Werden weiter den Stellen der Kodewörter $\Phi_X(x)$ bzw. $\Phi_Y(y)$ bzw. $\Phi_Z(z)$ die Aussagenvariablen e_i bzw. r_j bzw. p_k zugeordnet, dann können wir die Funktionen f_k bzw. g_j durch die aussagenlogischen Formeln B_k bzw. C_j repräsentieren, die bei den den Kodewörtern entsprechenden Variablenbelegungen genau dann den Wert 1 (wahr) annehmen, wenn die k-te bzw. j-te Stelle im Kodewort für f (x, z) bzw. g (x, z) den Wert 1 hat. Jedem Kodewort f (x) bzw. f(z) entspricht dann eine Elementarkonjunktion E_Φ (x) in den Variablen e_i bzw. $E_\Phi(z)$ in den Variablen p_k , die genau bei der dem Kodewort zugeordneten Variablenbelegung den Wert 1 annimmt. Damit ergeben sich die Formeln

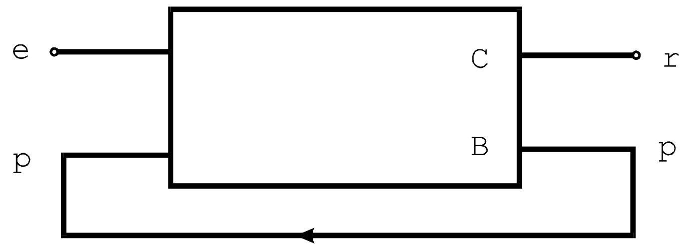
$$B_k (e, p) = \bigvee_{\Phi_Z(f (x, z))_k = 1} E_\Phi (x) E_\Phi (z) ,$$

$$C_j (e, p) = \bigvee_{\Phi_Y(g (x, z))_j = 1} E_\Phi (x) E_\Phi (z) ,$$

wobei e bzw. p die Variablentupel e_i bzw. p_k bezeichnen.
Wenn das Anweisungszeichen : = eine Zuordung mit Taktverzögerung bezeichnet, dann kann bei der gegebenen Kodierung f der Automat durch das Aweisungssystem

$$p_k := B_k (e, p) \qquad (1 \le k \le l (Z))$$
$$r_j := C_j (e, p) \qquad (1 \le j \le l (Y))$$

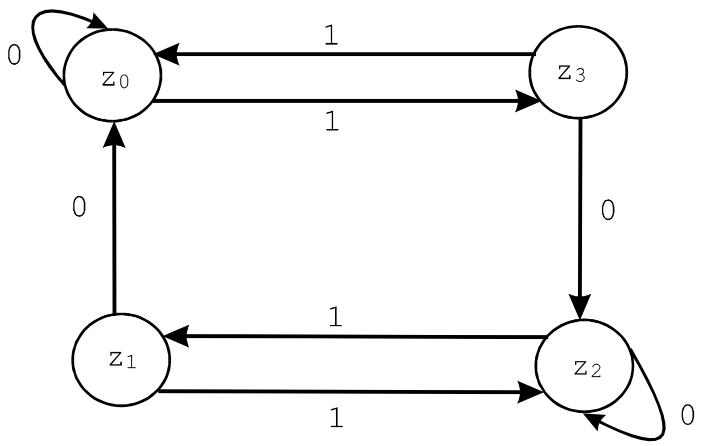
beschrieben werden.
Die Formeln B_k und C_j sind jetzt unter Benutzung geeigneter Schaltelemente (Gatter) durch Schaltnetze zu realisieren. Unter Verwendung von Verzögerungsleitungen um einen Takt (Speicher) ist damit der Automat durch ein Schaltwerk (etwa nach dem Modell von Huffmann) in der folgenden Weise beschreibbar:



Verzögerungsleitung

Beispiel: Zum Automaten nach gegebener Tafel bzw. Transitionsdiagramm

(f, g)	z_0	z_1	z_2	z_3
x_0	(z_0, y_0)	(z_0, y_0)	(z_2, y_0)	(z_2, y_0)
x_1	(z_3, y_1)	(z_2, y_1)	(z_1, y_0)	(z_0, y_0)



und der Kodierung Φ mit $\Phi_X(x_0) = 0$, $\Phi_X(x_1) = 1$, $\Phi_Y(y_0) = 0$, $\Phi_Y(y_1) = 1$, $\Phi_Z(z_0) = 00$, $\Phi_Z(z_1) = 01$, $\Phi_Z(z_2) = 10$, $\Phi_Z(z_3) = 11$

erhalten wir die Booleschen Funktionen f_1, f_2 und g_1 mit folgenden Wertetabellen:

e	p_1	p_2	f_1	f_2	r_1	e	p_1	p_2	f_1	f_2	r_1
0	0	0	0	0	0	1	0	0	1	1	1
0	0	1	0	0	0	1	0	1	1	0	1
0	1	0	1	0	0	1	1	0	0	1	0
0	1	1	1	0	0	1	1	1	0	0	0

Als Formeln entstehen

$$B_1 = (\bar{e}\ p1\ \overline{p2}) \vee (\bar{e}\ p1\ p2) \vee (e\ \overline{p1}\ \overline{p2}) \vee (e\ \overline{p1}\ p2) \underset{w}{=} (\bar{e}\ p1) \vee (e\ \overline{p1})$$

$$B_2 = (e\ \overline{p1}\ \overline{p2}) \vee (e\ p1\ \overline{p2}) \underset{w}{=} (e\ \overline{p2})$$

$$C = (e\ \overline{p1}\ \overline{p2}) \vee (e\ \overline{p1}\ p2) \underset{w}{=} (e\ \overline{p1})$$

($\underset{w}{=}$ Zeichen für Gleichwertigkeit)

Der Automat kann also durch das Anweisungssystem

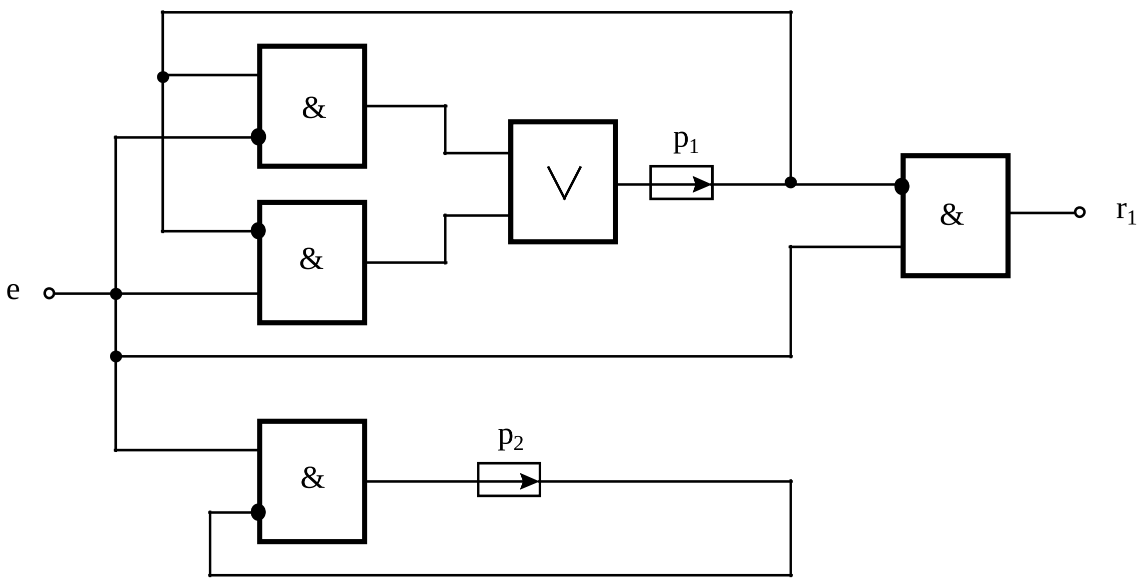
$$p1 := (\overline{e}\ p1) \vee (e\ \overline{p1})$$

$$p2 := (e\ \overline{p2})$$

$$r1 := (e\ \overline{p1})$$

beschrieben werden.

Das entsprechende Schaltwerk mit den Gattern & für die Konjunktion bzw. $\vee$ für die Alternative bzw. $\rightarrow$ für die Taktverzögerung (punktierte Eingänge bedeuten Negation) ergibt sich dann zu:



2.1 Semiotische Grundbegriffe

Formale Sprachen sind Mengen von Zeichenketten (Wörter) über endlichen Grundmengen (Alphabete). Durch Verkettung von Wörtern bzw. Wortmengen (Sprachen) können neue Wörter bzw. Wortmengen (Sprachen) erzeugt werden. Diese Verkettung $\circ$ ist eine assoziative Operation und die Menge aller Wörter über einer bestimmten Grundmenge bildet zusammen mit dieser vollständigen binären Operation eine *Halbgruppe*. Außerdem hatten wir das leere Wort ε (Zeichenkette der Länge 0) als neutrales Element eingeführt. Damit bildet die Menge aller Wörter mit der Verkettung als Operation und dem leeren Wort als Einselement ein *Monoid*.

Wie bei binären Operationen üblich, lassen sich Potenzen wie folgt erklären: Für jedes Wort p des Monoids sei $p^0 = \varepsilon$ und $p^{n+1} = p^n \circ p$ (n nat. Zahl). Die so eingeführten Potenzen sind vollständige unäre Operationen auf der Menge aller Wörter.
Analog lassen sich mit Hilfe der auf Wortmengen (Sprachen) L_1 und L_2 erweiterten Verkettung $L_1 \circ L_2 = \{ p \circ q \mid p \in L_1 \text{ und } q \in L_2 \}$ Potenzen von Sprachen L einführen nach: $L^0 = \{ \varepsilon \}$ und $L^{n+1} = L^n \circ L$ für beliebige natürliche Zahlen n.

Wegen der Assoziativität der Verkettung gelten für Wort- und Sprachpotenzen die üblichen Potenzgesetze: $p^n \circ p^m = p^{n+m}$, $(p^n)^m = p^{n \cdot m}$ und $L^n \circ L^m = L^{n+m}$, $(L^n)^m = L^{n \cdot m}$.

Für jede Sprache L wird ein Komplexprodukt (Stern von L) $L^* = \bigcup_{n \geq 0} L^n$ eingeführt.

Neben dem Stern wird häufig auch $L^+ = \bigcup_{n \geq 1} L^n$ mit $L^* = L^+ \cup \{\varepsilon\}$ benutzt.

Für die Sternbildung gelten die Hülleneigenschaften, d.h., es gilt :

$L \subseteq L^*$ (Einbettung),
aus $L_1 \subseteq L_2$ folgt $L_1^* \subseteq L_2^*$ (Monotonie) und

$(L^*)^* = L^*$ (Abgeschlossenheit) ,

außerdem ist L^* stabil, d.h. $L^* \circ L^* = L^*$ und damit die Verkettung auf L^* eine vollständige Operation.
L^* bildet also bzgl. dieser Verkettung ein Monoid, das $\{\varepsilon\} = L^0$ als Einselement besitzt. Man bezeichnet L^* als das von L erzeugte Monoid.

***Definition*:** Freies Monoid

Ein Monoid $(M, \circ, \varepsilon)$ mit der Trägermenge M, der assoziativen Verkettungsoperation $\circ$ und dem Einselement ε heißt frei, wenn eine nichtleere Teilmenge $B \subseteq M \setminus \{\varepsilon\}$ von M existiert und es zu jedem von ε verschiedenen Wort $p \in M$ genau eine nat. Zahl $n \geq 1$ und ein Tupel $(b_1, b_2, ..., b_n)$ von Elementen aus B gibt, so daß gilt: $p = b_1 \circ b_2 ... \circ b_n$.
B heißt die Basis (Erzeugendensystem) des Monoids und das Monoid erzeugbar durch B, d.h. $M = B^*$. Wenn B endlich, so heißt das Monoid endlich erzeugbar.
Künftig gehen wir von einer Basis B (Alphabet) als endliche Menge von Grundzeichen aus und bezeichnen das daraus erzeugte Monoid bei der eingeführten Verkettung durch B^* (Wortmenge über B). Zu jedem Monoidelement (Wort) p gibt es genau eine nat. Zahl l(p) mit $p \in B^*$, die Länge von p heißt und die Anzahl der Stellen in p angibt, an denen Elemente (Buchstaben) aus B stehen. Für alle Wörter p, q aus B gilt: l(pq) = l(p) + l(q), d.h., die Länge ist additiv.

Über freie Monoide $(B^*, \circ, \varepsilon)$ mit der Basis B gilt der

Satz: a) Aus $n \neq m$ folgt $B^n \cap B^m = \emptyset$ für alle nat. Zahlen $n, m > 0$;
b) B ist die einzige Basis von $(B^*, \circ, \varepsilon)$;
c) $px \neq \varepsilon$, $yq \neq \varepsilon$, und aus $px = qy$ folgt $p = q$ und $x = y$ für alle $x, y \in B$ und $p,q \in B^*$.

(Das Einselement ε ist nicht erzeugbar. Aus der Eigenschaft c) kann umgekehrt die Freiheit des Monoids geschlossen werden.)

Unter einem *Homomorphismus* vom Monoid $(M, \circ)$ auf das Monoid $(N, \lozenge)$ versteht man eine operationstreue Abbildung Φ von M auf N mit $\Phi(p \circ q) = \Phi(p) \lozenge \Phi(q)$ für alle p, q aus M. Dabei wird dem Einselement aus M das Einselement aus N zugeordnet. Wenn die Abbildung Φ eineindeutig ist, dann spricht man von einem *Isomorphismus*.

Satz: Das Monoid $(M, \circ)$ ist frei über B genau dann, wenn für jedes Monoid $(N, \lozenge)$ und jede (eindeutige) Abbildung Φ von B in N ein (eindeutig bestimmter) Homomorphismus $\tilde{\Phi}$ von $(M, \circ)$ auf $(N, \lozenge)$ existiert, mit $\tilde{\Phi}(b_1, \ldots b_n) = \Phi(b_1) \ldots \Phi(b_n)$ für alle $n \geq 1$ und b_i aus B.

($\tilde{\Phi}$ heißt homomorphe Fortsetzung von Φ, die im Existenzfall eindeutig bestimmt ist und für die $\tilde{\Phi} / B = \Phi$ gilt.)

Definition: Teilwort (Infix), Anfangswort (Präfix), Endwort (Postfix)
Sei B^* die Wortmenge über B und p,q Elemente aus B^* .
a) p heißt Teilwort von q genau dann, wenn Wörter u,v aus B^* mit q=upv existieren.
b) p heißt Anfangswort von q genau dann, wenn ein Wort v aus B^* mit q=pv existiert.
c) p heißt Endwort von q genau dann, wenn ein Wort u aus B^* mit q=up existiert.
(Wenn $p, q \neq \varepsilon$ und $p \neq q$, dann spricht man von einem echten Teil- bzw. Anfangs- bzw. Endwort.)

Für das Arbeiten mit Wörtern bzw. Sprachen betrachten wir weiter die einstellige Operation Einsetzung und die vierstellige Relation Ersetzung.

Definition: a) Zu jedem Elementepaar (b,q) mit b aus B und q aus B^* ist eine durch b/q bezeichnete einstellige Operation (Ersetzung von q für b) bestimmt:

$$p(b/q) = \begin{cases} p & \text{falls } m=0 \text{ oder } p=\varepsilon \\ u_o q u_1 q \ldots u_m & \text{falls } p=u_o b u_1 b \ldots u_m,\ u_i \in (B \backslash b)^* \text{ und } m>0. \end{cases}$$

(Das Ergebnis ist eindeutig, da die Zerlegung $u_o b u_1 b \ldots u_m$ von p eindeutig ist, und wird als das durch Einsetzen von q für b in p entstandene Wort bezeichnet.)

b) Zu jedem Tupel von Elementepaaren (b_i, q_i) mit b_i aus B und q_i aus B* ist eine durch $b_1/q_1 \ldots b_n/q_n$ bezeichnete einstellige Operation (<u>simultane Einsetzung</u> von q_i für b_i) bestimmt:

$$p(b_1/q_1 \ldots b_n/q_n) = \begin{cases} p & \text{falls } m=0 \text{ oder } p=\varepsilon \\ u_0 q_i\, u_1 q_i \ldots u_m & \text{falls } p=u_0 b_{i_1} u_1 b_{i_2} \ldots u_m,\ m>0, \\ & 1 \le i_1,\ldots,i_m \le n \text{ und} \\ & u_i \in (B\backslash\{b_1,\ldots,b_n\})^*. \end{cases}$$

(Analog zu a), ist das Ergebnis wieder eindeutig und wird als das durch <u>simultanes Einsetzen von</u> q_i <u>für</u> b_i <u>in</u> p entstandene Wort bezeichnet.)

Beispiel: Mit dem Alphabet B={x,y,z,(,),+,*} und dem Wort p=((x+y)*z) entsteht durch simultanes Einsetzen von x/x*z, y/(y*z, */ε, z/ε das Wort ((x*z)+(y*z)), Die Folge (einfacher) Einsetzungen liefert dagegen das Wort ((x)+(y)).

<u>*Bemerkung*</u>: Simultane Einsetzungen können nach Umbenennung der Elemente von B, für die eingesetzt wird, durch Folgen einfacherer Einsetzungen erzeugt werden. Im obigen Beispiel entsteht nach Umbenennung von x bzw. y bzw. * bzw. z in a bzw. b bzw. $\circ$ bzw. c aus ((a+b)$\circ$c) durch die Folge von Einsetzungen a / x*z), b/(y*z, $\circ$/ε und c/ε wieder das Wort ((x*z)+(y*z)).

Definition: Die Wörter p, u, v, q aus B^* stehen in der Relation Ers (Ersetzung; in Zeichen : Ers(p,u,v,q)) genau dann, wenn eine nat. Zahl n und Zerlegungen $p = p_0 p_1 \ldots p_{2n}$ und $q = q_0 q_1 \ldots q_{2n}$ existieren, so daß $p_{2i} = q_{2i}$ für alle $i \ge 0$ und falls n>0 ist $p_{2i-1} = u$, $q_{2i-1} = v$ für alle $i \ge 1$. (D.h., aus p entsteht q, wenn an keiner oder an beliebigen Stellen, an denen in p das Teilwort u vorkommt, dieses dort durch v "ersetzt" wird.)

Beispiel: Für B={a,b,c,d}, p=abcdcbd, u=bcb und v=bd gilt Ers(p,u,v,abcbdd) mit p=abcud und Ers(p,u,v,abdcbd) mit p=aucbd.

<u>*Bemerkung*</u>: Wie im Beispiel zu sehen, handelt es sich bei der Ersetzung um eine Relation, da das Wort q durch u,v,p nicht eindeutig bestimmt wird. Offensichtlich gilt immer: Ers(p,u,v,p); Ers(p,p,v,v); Ers(p,u,u,p).

Häufig wird eine weitere als <u>Spiegelung</u> ~ bezeichnete einstellige Operation für Wörter benutzt, die durch $\tilde{\varepsilon} = \varepsilon$ und $\widetilde{pq} = q\tilde{p}$ für alle $q \in B$ und $p \in B^*$ definiert ist. Diese Operation besitzt offensichtlich die Eigenschaften $l(p) = l(\tilde{p})$ (längentreu), $\tilde{\tilde{p}} = p$ (involutorisch) und $\widetilde{pq} = \tilde{q}\tilde{p}$ (antihomomorph bzgl. Verkettung) für alle p,q $\in B^*$.

2.2 Regelgrammatiken und Chomsky-Klassifikation

Zur Beschreibung formaler Sprachen als Teilmengen der Wortmenge über einem gegebenen Alphabet werden Regelsysteme verwendet, durch die die Erzeugung von Wörtern der Sprache ermöglicht wird. Bei diesem Definitionsprinzip wird von algebraischen Strukturen der Form (A,R) mit einer nichtleeren abzählbaren (endlichen) Menge A und einer zweistelligen Relation R über A^* ausgegangen, die Semi-Thue Systeme genannt werden. Dafür wird ein Ableitungsbegriff wie folgt eingeführt:

- Bezeichnen p,q Wörter aus A^*, dann heißt q direkt ableitbar aus p (in Zeichen: $p \rightarrow q$), wenn ein Wortpaar (Regel) r=(u,v) aus R und Wörter p_1, p_2 aus A^* existieren, so daß $p=p_1up_2$ und $q=p_1vp_2$.
- q heißt ableitbar aus p, wenn $p \Rightarrow q$ wobei $\Rightarrow$ die reflexiv, transitive Hülle von $\rightarrow$ bezeichnet.
- Jeder Teilmenge P von A^* kann damit eine als Ableitungsmenge bezeichnete Menge $Abl(P) = \{ q \mid p \Rightarrow q\} \subseteq A^*$ zugeordnet werden.

Diese Operation für Teilmengen P, Q von A ist offenbar eine Hüllenoperation mit den Eigenschaften $P \subseteq Abl(P)$, aus $P \subseteq Q$ folgt $Abl(P) \subseteq Abl(Q)$, $Abl(Abl(P)) \subseteq Abl(P)$.
Zur Erzeugung formaler Sprachen führen wir nach Chomsky den Grammatikbegriff ein.

Definition a): (Regel-)Grammatik

Eine algebraische Struktur G=(M, A, R, S) heißt eine Grammatik über A, wenn
- M, A nichtleere disjunkte endliche Mengen,
- $R \subseteq Z^* x\, Z^*$ endlich mit Z=M v A und
- $S \in M$.

Bezeichnungen: M heißt die Menge der Metazeichen (Variable, Nichtterminale), A heißt die Menge der Alphabetzeichen (Konstante, Terminale), R heißt die Regelmenge und ihre Elemente (u,v) Regeln (Produktionen) und S wird Startsymbol (Axiom) genannt. Statt (u,v) werden die Regeln auch in der Form $u \rightarrow v$ geschrieben. Falls keine Verwechslungen mit Mengenbezeichnungen zu befürchten sind, sollen Metazeichen durch Großbuchstaben gekennzeichnet werden.

Unter Verwendung des oben angegebenen Ableitungsbegriffs erzeugen wir mit Hilfe der Grammatik eine formale Sprache nach

Definition b): (Regel-)Sprache

Die Menge $L(G)=Abl(\{S\}) \cap A^* = \{p \mid p \in A^* \text{ und } S \Rightarrow p\}$ heißt die von der Grammatik G erzeugte Sprache.

L(G) enthält alle mit Hilfe der Regelmenge R aus dem Startsymbol S ableitbaren Wörter über A. Bei Vorgabe der Grammatik G ist die von ihr erzeugte Sprache eindeutig definiert. Umgekehrt kann eine Sprache von verschiedenen Grammatiken erzeugt werden. Solche Grammatiken nennen wir äquivalent.

Definition c): Die Grammatiken G und G' heißen äquivalent, wenn sie die gleichen Sprachen erzeugen: L(G)=L(G').

Beispiel: 1. Gegeben sei die Grammatik G=({S},{a,b}, {(S,aSb),(S,ab)},S). G erzeugt die Sprache L(G)={$a^n b^n$ | $n \geq 1$ nat.Zahl}. Die Erzeugung der Sprache L(G) durch die Regeln von G veranschaulicht der Graph

$$
\begin{tikzcd}
S \arrow[r] \arrow[d] & aSb \arrow[r] \arrow[d] & aaSbb \arrow[r, Rightarrow] \arrow[d] & a^{n-1}Sb^{n-1} \arrow[d] \\
ab & a^2b^2 & a^3b^3 & a^nb^n
\end{tikzcd}
$$

(Der entsprechende Nachweis ist induktiv über die Länge der entsprechenden Wörter zu führen.)

(Häufig werden von der Grammatik nur die Regeln angegeben, wobei S immer das Startsymbol bezeichnet und Regeln mit gleichen linken Seiten in der Form $u \rightarrow v_1 \mid v_2 ... \mid v_m$ geschrieben werden.)

2. Durch die Grammatik G mit den Regeln (1) $S \rightarrow LaK$, (2) $aK \rightarrow WbbK$, (3) $aW \rightarrow Wbb$, (4) $LWb \rightarrow LaB$, (5) $LWb \rightarrow aB$, (6) $Bb \rightarrow aB$, (7) $BK \rightarrow K$, (8) $BK \rightarrow \varepsilon$ (ε leeres Wort) wird die Sprache L(G)={a^{2^n}| $n \geq 1$ nat.Zahl} erzeugt. Wie bemerkt, kann der Beweis dieser Behauptung induktiv geführt werden, wobei die entsprechenden Induktionsschritte durch die folgenden Graphen verdeutlicht werden. (An den Pfeilen stehen die anzuwendende Regel und bei $\Rightarrow$ die Länge der Ableitung.)

$$
\begin{tikzcd}
S \arrow[r, "(1)"] & LaK \arrow[r, "(2)"] & LWbbK \arrow[r, "(5)"] \arrow[d, "(4)"] & aBbK \arrow[r, "(6)"] & aaBK \arrow[r, "(8)"] \arrow[d, "(7)"] & aa \in L(G) \\
 & & LaBbK \arrow[d, "(6)"] & & aaK \arrow[d, "(2)"] & \\
 & & LaaBK \arrow[r, "(8)"] \arrow[d, "(7)"] & Laa & aWbbK \arrow[d, "(3)"] & \\
 & & LaaK \arrow[d, "(2)"] & & WbbbbK & \\
 & & LaWbbK \arrow[r, "(3)"] & LWbbbbK \arrow[r] & ... &
\end{tikzcd}
$$

Nur das Wort LWb^4K ist weiter ableitbar. Für $LWb^{2^n}K$ entsteht im Induktionsschritt der Graph:

$$
\begin{tikzcd}
LWb^{2^n}K \arrow[r, "(5)"] \arrow[d, "(4)"] & aBb^{2^n-1}K \arrow[r, Rightarrow, "(6)", "2^n-1"'] & a^{2^n}BK \arrow[r, "(8)"] \arrow[d, "(7)"] & a^{2^n} \in L(G) \\
LaBb^{2^n-1}K \arrow[d, Rightarrow, "(6)", "2^n-1"'] & & a^{2^n}K \arrow[d, "(2)"] & \\
La^{2^n}BK \arrow[r, "(8)"] \arrow[d, "(7)"] & La^{2^n} & a^{2^n-1}WbbK \arrow[d, Rightarrow, "(3)", "2^n-1"'] & \\
La^{2^n}K \arrow[d, "(2)"] & & Wb^{2^{n+1}}K & \\
La^{2^n-1}WbbK \arrow[r, Rightarrow, "(3)", "2^n-1"'] & LWb^{2^{n+1}}K \arrow[r] & ... &
\end{tikzcd}
$$

In Abhängigkeit von der Gestalt der Regeln, kennzeichnete Chomsky verschiedene Typen von Grammatiken und Sprachen.

Definition**:** Chomsky-Typen

- Eine Grammatik G=(M,A,R,S) heißt Typ-i Grammatik, wenn für ihre Regeln r=(u,v) folgendes gilt:
  a) Typ-0: Es existiert ein $m \in M$ und Wörter p_1, p_2, $q \in Z^*$ mit $u=p_1mp_2$ und $v=p_1qp_2$.
  b) Typ-1 monoton : Es gilt immer $0 < l(u) \leq l(v)$.
  kontextabhängig : Es existiert ein $m \in M$ und Wörter $p_1,p_2 \in M^*$ und $q \in Z^+$ mit $u=p_1mp_2$ und $v=p_1qp_2$.
  (Bemerkung: Häufig wird auch die Regel (S,ε) zugelassen, dann aber für alle anderen $v \in (Z\backslash S)^+$ gefordert. Jede kontextabhängige Grammatik ist monoton.)
  c) Typ-2 kontextfrei : Es gilt immer $u \in M$ und $v \in Z^*$.
  d) Typ-3 rechtslinear : Es gilt immer $u \in M$ und $v \in A^* \vee A^*M$.
  linkslinear : Es gilt immer $u \in M$ und $v \in A^* \vee MA^*$.
  (Bemerkung: Grammatiken vom Typ-3 heißen auch linear.)
- Eine formale Sprache $L \subseteq A^*$ heißt Typ-i Sprache, wenn es eine sie erzeugende Typ-i Grammatik G gibt, mit L=L(G).

(*Bemerkung*: Im konkreten Fall spricht man auch von monotonen, kontextabhängigen, kontextfreien, rechtslinearen, linkslinearen bzw. linearen Sprachen. Da eine Sprache von verschiedenen Grammatiken erzeugt werden kann, ist der Typ der Sprache nicht eindeutig festgelegt, sondern nur ihr Maximaltyp.)
Die vom Typ-0 Grammatiken erzeugten Sprachen werden auch rekursiv aufzählbar genannt. Sprachen vom Typ-3 heißen auch regulär (Vergl. Kap. 1.2.).

Eigenschaften:

- Jede Grammatik vom Typ-i mit i>0 ist auch eine Typ-0 Grammatik.
- Die Grammatiken vom Typ-3 sind auch Typ-2 Grammatiken.
- Es gibt Typ-1 Grammatiken, die keine Typ-2 Grammatiken sind und umgekehrt.
- Es gibt Typ-1 bzw. Typ-2 Grammatiken die nicht vom Typ-3 sind.
- Sprachen vom Typ-i mit $i > 0$ sind entscheidbare Mengen, während Sprachen vom - Maximaltyp 0 nicht entscheidbar sein können.
- Alle endlichen Sprachen sind regulär.
- Es gibt Sprachen, die sowohl vom Typ-2, als auch vom Typ-1 sind.

Die Gesamtheit aller Sprachen von einem Typ-i soll im weiteren durch L_i bezeichnet werden.
Bezüglich der erzeugten Sprachen interessieren lediglich die Äquivalenzklassen der Grammatiken. Unter den Grammatiken einer Äquivalenzklasse, die die gleichen Sprachen erzeugen, können wir uns auf bestimmte *Normalformen* als Repräsentanten beziehen.

Satz**:** Normalformgrammatiken

Zu jeder Typ-0 Grammatik gibt es eine äquivalente Typ-0 Grammatik, deren Regeln die Form $xy \to xz$ oder $xy \to zy$ oder $x \to yz$ oder $x \to a$ oder $x \to \varepsilon$ mit $x, y, z \in M$ und $a \in A$ haben.
(Diese als Normalform bezeichnete Grammatik ist effektiv herstellbar. Bei der entsprechenden Transformation bleibt Monotonie bzw. Kontextabhängigkeit bzw. Kontextfreiheit der Grammatik erhalten.)

Hilfssätze: a) Zu jeder Typ-0 Grammatik gibt es eine äquivalente Grammatik, deren Regeln (u,v) die Form $u \in M^+$ haben, also Wörter aus Metazeichen sind. (Dazu werden Kopien a' der Alphabetzeichen a zur Metazeichenmenge hinzugenommen, danach in den Regeln die Alphabetzeichen durch ihre Kopien ersetzt und zu der so entstandenen Regelmenge die Regeln (a, a') für alle a aus A hinzufügt. Die Regeln der so entstehenden äquivalenten Grammatik besitzen die Eigenschaft $u \in M^+$ und $v \in M^*$ oder $u \in M$ und $v \in A$. Dabei bleiben die Typen 0, 1 und 2 erhalten, der Typ 3 nicht.)

b) Zu jeder Typ-0 Grammatik gibt es eine äquivalente Grammatik, deren Regeln (u, v) die Form $u, v \in M^2$ oder $u \in M$ und $v \in M^2 \cup M \cup A \cup \{\varepsilon\}$ haben. (Diese Form kann durch Einführung neuer Metazeichen und neuer Regeln sogar in einer weiteren Spezialisierung erreicht werden.)

***Satz*: ** Jede von einem endlichen Automaten A akzeptierte Sprache L(A) kann von einer rechtslinearen Grammatik G erzeugt werden, d.h. L(A) = L(G).

Beweis: $A = (Z, X, f, z_0, F)$ $z_0 \in Z$, $F \subseteq Z$ Finalzustände

Annahme $z_0 \notin F$:

Konstruktion der Grammatik $G = (Z, X, R, z_0)$ mit $R = \{(z, xf(x,z)), (z',x) \mid f(x,z') \in F\}$

Man kann zeigen: $f(p, z) = z' \leftrightarrow z \xrightarrow[R]{*} pz'$

$L(A) \subseteq L(G)$ $px \in L(A)$, $f(p, z_0) = z' \rightarrow z_0 \xrightarrow[R]{*} pz'$

$f(x, z') \in F \rightarrow (z', x) \in R$

$px \in L(G) \leftarrow$ $z_0 \xrightarrow[R]{*} px$

$L(G) \subseteq L(A)$ $px \in L(G)$, $z_0 \xrightarrow[R]{*} px$

$z_0 \xrightarrow[R]{*} pz' \xrightarrow[R]{} px$, $z' \in Z$

$px \in L(A) \leftarrow f(p, z_0) = z', f(x, z') \in F$

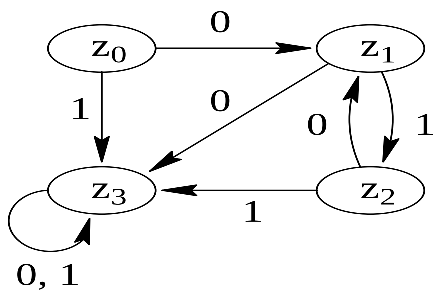
Annahme $z_0 \in F$: $\rightarrow e \in L$, $L(G) = L \setminus \{e\}$

Übergang von der Grammatik G zu G´: G´: $s \notin Z$, $R' = R \cup \{(s, z_0); (s,e)\}$

$\rightarrow$ L(A) = L(G´)

Beispiel:

A:



G: $R = \{ (z_0, 0z_1); (z_0, 1z_3); (z_0, 0);$
$(z_1, 0z_3); (z_1, 1z_2);$
$(z_2, 0z_1); (z_2, 1z_3); (z_2, 0);$
$(z_3, 0z_3); (z_3, 1z_3) \}$

L(A) = (10)* = L(G)

Satz: Jede von einer (rechts-)linearen Grammatik G erzeugte Sprache L(G) wird von einem endlichen Automaten A akzeptiert.

Beweis: $G = (M, X, R, S)$

Konstruktion eines nichtdeterministischen endlichen Automaten

$A = (Z, X, f, [s], [e])$ mit $[s] \in Z$ Startzustand
$[e] \in Z$ Finalzustand

und $f(e, [m]) = \{[q] \mid (m, q) \in R\}$ $m \in M$; $[m], [q] \in Z$
$f(x, [xq]) = \{[q]\}$ $x \in X, q \in X^* \cup X^* M$; $[q], [xq] \in Z$

Behauptung:

$[r] \in f(w, [s])$ gdw. $S \xrightarrow{*} pm \rightarrow pqr$ mit $(m, qr) \in R$ und $w = pq$
oder $m = S$ und $w = e$

(Induktion über Ableitungslänge)

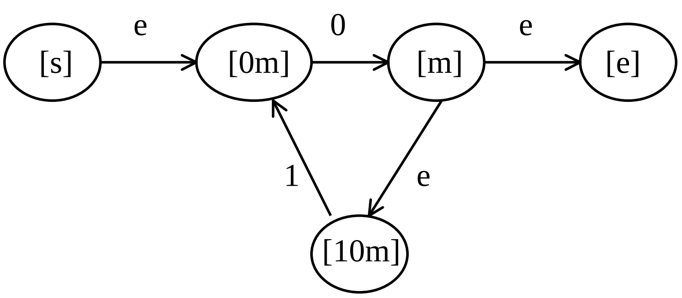
$\rightarrow$ $w \in L(A)$ gdw. $S \xrightarrow{*} pm \rightarrow w$ für $m \in M$; $p \in X^*$

Beispiel: R: $\{(s, 0m); (m, 10m); (m, e)\}$; $L(G) = 0(10)^*$

A:

	0	1	e
[s]	∅	∅	[0m]
[0m]	[m]	∅	∅
[m]	∅	V	[e]
[10m]	∅	[0m]	∅
[e]	∅	∅	∅

$L(A) = 0(10)^*$



Bemerkung: Nichtdeterministische endliche Automaten mit e-Übergängen akzeptieren dieselbe Sprachklasse wie deterministische endliche Automaten.

***Satz*:** Jede durch eine rechts-lineare Grammatik G erzeugte Sprache L(G) ist regulär.

Beweis: $G = (M, A, R, s)$, $M = \{m_1, \dots m_n\}$, $s = m_1$

$R_k \subseteq R$ Menge aller Regeln, die nur Metazeichen $m_1, \dots, m_k$ verwenden.

$W_{ijk} = \{p \mid m_i \xrightarrow[R_k]{*} pm_j\} \subseteq A^*$

$X_j = \{p \mid (m_j, p) \in R\} \subseteq A^*$ endlich

$\rightarrow$ $W_{ij0} = \{p \mid (m_i, pm_j) \in R\}$ endlich und

$W_{ijk-1} \subseteq W_{ijk} = W_{ijk-1} \cup Wi_{kk-1} \cdot W^*_{kkk-1} \cdot W_{kjk-1}$ und

$W_{ijn} = \{p \mid m_i \xrightarrow[R]{*} pm_j\}$

Daraus folgt W_{ij0} regulär; X_j regulär; und wenn W_{ijk-1} regulär, dann auch W_{ijk} regulär. Insgesamt also ist

$$L(G) = \bigcup_{j=1}^{n} W_{1jn} \cdot X_j \cup X_1 \text{ regulär}$$

Beispiel:

$n = 2$; $G = (\{S, m\}, \{0, 1\}, R, s)$; $R = \{(S, 0m), (m, 10m), (m, e)\}$

$W_{110} = W_{210} = \varnothing$, $W_{120} = 0$, $W_{220} = 10$ $(m_1 = S, m_2 = m)$

$X_1 = \varnothing$, $X_2 = e$

$W_{121} = W_{120} \cup W_{110} \cdot W^*_{110} \cdot W_{120} = 0$

$W_{221} = W_{220} \cup W_{210} \cdot W^*_{110} \cdot W_{120} = 10$

$W_{122} = W_{121} \cup W_{121} \cdot W^*_{221} \cdot W_{221} = 0 \cup 0(10)^*(10) = 0(10)^*$

$L(G) = W_{112} \cdot X_1 \cup W_{122}X_2 \cup X_1 = W_{122} = 0(10)^*$

***Satz*:** Zu jeder regulären Sprache L existiert eine sie erzeugende Typ-3 Grammatik (rechts-linear) G_L.

Beweis: Wenn L endlich: $L = \{p_1, \dots, p_n\}$ $G_L = (M, A, R, S)$ rechts-linear mit $M = \{S\}$, $R = \{(S, p_i) \mid 1 \le i \le n\}$ und $L = L(G_L)$ d.h. Elementarsprachen sind vom Typ-3.

Sei L_i vom Typ-3 (rechts-linear) und $G_i = (M_i, A, R_i, S_i)$ rechts linear mit $M_i \cap M_j = \varnothing$ für $i \ne j$ und $L_i = L(G_i)$

$\Rightarrow$ a) $L_1 \cup L_2$ rechts-linear mit $L(G_{12}) = L_1 \cup L_2$ für
$G_{12} = (M, A, R, S)$, $M = M_1 \cup M_2 \cup \{S\}$, $S \notin M_1 \cup M_2$ und
$R = R_1 \cup R_2 \cup \{(S, S_1)(S, S_2)\}$

b) $L_1 \cdot L_2$ rechts-linear mit $L(G_{12}) = L_1 \cdot L_2$ für
$G_{12} = (M, A, R, S_1)$, $M = M_1 \cup M_2$, $R = R_1{}^{(m, p)}/_{(m, pS2)} \cup R_2$, $m \in M_1$, $p \in A^*$

c) L_0^* rechts-linear mit $L(G_*) = L_0^*$ für
$G_* = (M_0, A, R, S_0)$, $R = R_0{}^{(m, p)}/_{(m, pS0)} \cup \{S_0, e\}$, $m \in M_0$, $p \in A^*$

Insgesamt sind demnach ausgehend von den Elementarsprachen alle regulären Sprachen rechts-linear.

Beispiel:

$L = 0(10)^*$ regulär mit $A = \{0, 1\}$

$\{0\} = L(G_0)$ mit $G_0 = (S_0, A, \{(S_0, 0)\}, S_0)$

$\{1\} = L(G_1)$ mit $G_1 = (S_1, A, \{(S_1, 1)\}, S_1)$

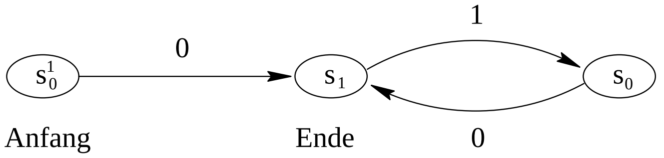
==> $\{10\} = L(G_{10})$ mit $G_{10} = (\{S_0, S_1\}, A, \{(S_0, 0)(S_1, 1S_0)\}, S_1)$

==> $\{(10)^*\} = L(G_{(10)*})$ mit $G_{(10)*} = (\{S_0, S_1\}, A, R_{(10)*}, S_1)$

$R_{(10)*} = \{(S_0, 0S_1), (S_1, 1S_0), (S_1, e)\}$

==> $L = L(G)$ mit $G = (\{S'_0, S_0, S_1\}, A, R, S'_0)$

$R = \{(S'_0, 0S_1), (S_0, 0S_1), (S_1, 1S_0)(S_1, e)\}$



Folgerung: Eine Sprache ist genau dann regulär, wenn sie (rechts-)linear ist.

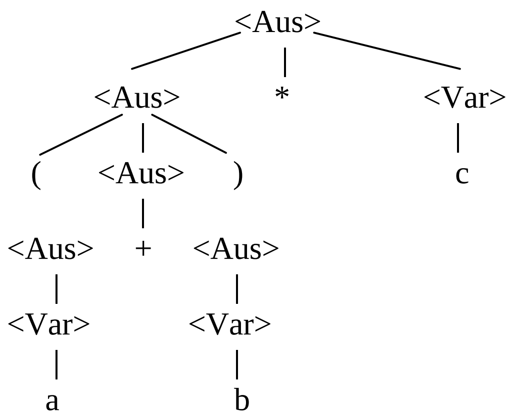
Bemerkung: Jede durch eine rechts-lineare Grammatik erzeugte Sprache kann durch eine links-lineare Grammatik erzeugt werden und umgekehrt.

(Spiegelung der rechten Seiten der Regeln $(m, p) \in R_{rechts} \leftrightarrow (m, \tilde{p}) \in R_{links}$)

Folgerung: Die Klasse der regulären Sprachen stimmt mit der Klasse der Typ-3 Sprachen überein.

2.3 Kontextfreie Grammatiken und Sprachen

Kontextfreie Grammatiken und Sprachen (Typ-2) werden häufig zur Definition der Syntax von Programmiersprachen, bei der Syntaxanalyse bzw. der Analyse von Blockstrukturen eingesetzt. Die Menge der Regeln (m, v_i) mit dem Metazeichen m auf der linken Seite werden dabei bevorzugt in der Backus-Naur-Form $m \rightarrow v_1 | v_2 \ldots | v_n$ notiert.
Als Metazeichen treten vielfach selbst wieder Wörter auf, die zur syntaktischen Kennzeichnung in spitze Klammern gesetzt werden.
Beispiel: Syntax der arithmetischen Ausdrücke ohne Konstanten
Terminalzeichen: a, b, ... , z, +, *, (,) ; Metazeichen: <Aus>, <Var>; Startzeichen: <Aus>
Regeln: <Aus> → <Aus> + <Aus> | <Aus> * <Aus> | (<Aus>) | <Var>
<Var> → a | b ... | z
Die Ableitung für das Wort (a + b) * c aus der von dieser kontextfreien Grammatik erzeugten Sprache kann wie folgt als Graph (*Ableitungsbaum*) beschrieben werden:



Sei G = (M, A, R, S) eine kontextfreie Grammatik mit den Regeln (m, v) $\in$ R und der erzeugten Sprache $L(G) = \{ p \mid p \in A^* \text{ und } S \Rightarrow p \} = \text{Abl}(S) \cap A^*$.

***Definition*: Ableitungsbaum**

Der geordnete bewertete Wurzelbaum W = (N, K, <, b) mit der endlichen Knotenmenge N, der Kantenmenge K, der irreflexiven linearen Ordnung < auf N und der Bewertung b der Knoten mit Zeichen aus $M \cup A \cup \{\varepsilon\}$ heißt Ableitungsbaum für das Wort $p \in A^*$ bzgl. der Grammatik G falls
- b (Wurzel) = S,
- wenn Nach (n) = $(n_1 \ldots n_k)$, dann gilt $(b(n), b(n_1) \ldots b(n_k)) \in R$ für alle $n \in N$,
- wenn n Blatt, dann ist $b(n) \in A \cup \{\varepsilon\}$, sonst $b(n) \in M$,
- wenn Rand (W) = $\{n_1 \ldots n_k\}$, dann ist $b(n_1) \ldots b(n_k) = p$.

(Nach (n) bzw. Rand(W) bezeichnet das Tupel der Nachfolgeknoten von n bzw. Blätter von W in der gegebenen Ordnung < , wo jeder Knoten n < jedem Nochfolger von n ist und die Nachfolgeknoten von links nach rechts geordnet sind.)

Satz: Beziehung zwischen Ableitungsbäumen und Ableitungen

Gegeben sei die kontextfreie Grammatik G = (M, A, R, S). Dann gilt:
$p \in L(G)$ genau dann, wenn ein Ableitungsbaum für p bzgl. G existiert.

Beweis: Allgemeiner kann induktiv über die Länge einer Ableitung bzw. über die Höhe des entsprechenden Ableitungsbaumes gezeigt werden, daß für alle $q \in A^*$ und $m \in M$ eine Ableitung $m \Rightarrow q$ genau dann existiert, wenn es einen Ableitungsbaum W mit b(Wurzel) = m und b(Rand(W)) = q gibt. Dabei ist zu beachten, daß ein Ableitungsbaum mehrere Ableitungen für ein Wort $p \in L(G)$ erzeugt, je nachdem in welcher Reihenfolge die den Unterbäumen des Ableitungsbaumes für p entsprechenden Ableitungen $m \Rightarrow q$ für Teilwörter von p angewendet werden.

Beispiel: Bezüglich der Grammatik G=({S,T}, {a, b}, R, S) mit R = { $S \rightarrow aTS \mid a$, $T \rightarrow SS \mid SbT \mid ba$ } entsteht für das Wort aabbaa $\in$ L(G) der folgende Ableitungsbaum :

```
      S
    / | \
   a  T  S
    / | \  \
   S  b  T  a
   |     / \
   a    b   a
```

Damit können u.a. die Ableitungen $S \rightarrow aTS \rightarrow aSbTS \rightarrow aSbTa \rightarrow aabTa \rightarrow aabbaa$,
$S \rightarrow aTS \rightarrow aTa \rightarrow aSbTa \rightarrow aSbbaa \rightarrow aabbaa$,
$S \rightarrow aTS \rightarrow aSbTS \rightarrow aabTS \rightarrow aabbaS \rightarrow aabbaa$
erzeugt werden. Die zweite bzw. dritte Ableitung ist dadurch ausgezeichnet, daß hier die Erzeugung des Wortes aabbaa entsprechend dem Aufbau des Ableitungsbaumes von rechts nach links bzw. von links nach rechts vorgenommen wird. Man spricht dann, von einer Rechts- bzw. Linksableitung.

Eigenschaften:

a) Zu jedem Ableitungsbaum existiert genau eine Rechts- und eine Linksableitung.

b) Existieren zu einem Wort aus L(G) mehrere Ableitungsbäume bzw. mehrere Rechts- / Linksableitungen, dann heißt G mehrdeutig.

c) Eine kontextfreie Sprache heißt ererbt mehrdeutig, wenn jede sie erzeugende kontextfreie Grammatik mehrdeutig ist.

d) Wenn W Ableitungsbaum bzgl. der kontextfreien Grammatik G ist und $l = Max(l(p) \mid (m,p) \in R)$ dann gilt $l(Rand(W)) \leq l^{Höhe\ (W)}$.

e) L(G) = { b(Rand(W)) | W Ableit.baum bzgl. G, b(Wurzel) = S und $b(Rand(W)) \in A^*$}

Definition: Reduzierte Grammatik

G = (M, A, R, S) sei eine kontextfreie Gramatik

a) $m \in M$ heißt produktiv, wenn ein $p \in A^*$ mit $m \Rightarrow p$ existiert.

b) $m \in M$ heißt erreichbar, wenn $p, q \in (M \cup A)^*$ mit $S \Rightarrow pmq$ existieren.

c) G heißt reduziert, wenn alle m aus M erreichbar und produktiv sind.

Satz: Existenz äquivalenter reduzierter kontextfreier Grammatiken

a) Zu jeder kontextfreien Grammatik G mit $L(G) \neq \emptyset$ existiert eine äquivalente kontextfreie Grammatik, die nur produktive Metazeichen besitzt.

b) Zu jeder kontextfreien Grammatik existiert eine äquivalente kontextfreie Grammatik, die nur erreichbare Metazeichen besitzt.

c) Zu jeder kontextfreien Grammatik G mit $L(G) \neq \emptyset$ existiert eine äquivalente kontextfreie reduzierte Grammatik.

Beweis: Bei a) und b) sind die äquivalenten Grammatiken nach Bildung der Teilmengen produktiver bzw. erreichbarer Metazeichen und dementsprechender Reduktion der Regelmengen effektiv konstruierbar. Werden erst die unproduktiven und danach die unerreichbaren Metazeichen entfernt, dann entsteht im Ergebnis eine äquivalente reduzierte Grammatik. (Vergleiche Bemerkung 3.)

Bemerkungen:

1) Das Metazeichen S ist produktiv genau dann, wenn $L(G) \neq \emptyset$.
2) Im Ableitungsbaum von $p \in L(G)$ bzgl. G ist jeder innere Knoton (nicht Blatt) mit einem produktiven und erreichbaren Metazeichen bewertet.
3) Es existieren produktive Metazeichen, die von S nur über unproduktive Zeichen erreichbar sind.
4) Für kontextfreie Grammatiken G ist $\varepsilon \in L(G)$ genau dann, wenn S in der Grammatik $(M \cup A, \emptyset, R, S)$ produktiv ist, d.h. die ε-Eigenschaft ist entscheidbar. (Allgemein ist für kontextfreie Grammatiken auch $m \Rightarrow \varepsilon$ entscheidbar. Ein Metazeichen m mit dieser Eigenschaft heißt nullierbar.)
5) Zu jeder kontextfreier Grammatik G existiert eine kontextfreie Grammatik G', die keine ε-Regel (m, ε) enthält und für die $L(G') = L(G) \setminus \{\varepsilon\}$ gilt.
6) Zu jeder kontextfreien Grammatik existiert eine äquivalente kontextfreie Grammatik, ohne Kettenregeln (u,v) mit $u,v \in M$ (Metazeichen werden durch Metazeichen ersetzt).

Die Grammatiken ohne ε-Regeln bzw. ohne Kettenregeln mit den in 5) bzw. 6) genannten Eigenschaften sind durch geeignete Manipulationen der Regelmengen effektiv herstellbar.

Definition: Chomsky-Normalform

Eine kontextfreie Grammatik G = (M, A, R, S) heißt in Chomsky-Normalform wenn für jede Regel $(u, v) \in R$ gilt : $u \in M$ und $v \in M^2 \cup A$.

<u>Satz</u>**:** <u>Konstruktion einer reduzierten Chomsky-Normalform</u>

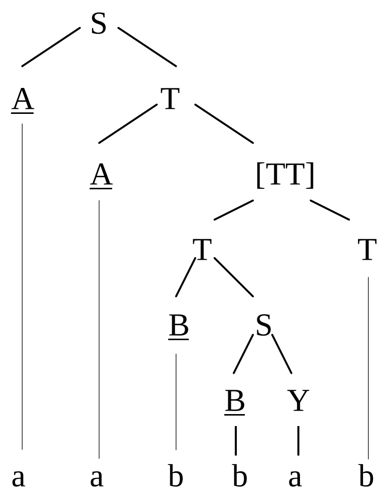
> Jede kontextfreie Sprache L mit $L \neq \emptyset$ und $\varepsilon \notin L$ kann durch eine kontextfreie reduzierte Chomsky-Normalform erzeugt werden.

Beweis: Nach den vorangegangenen Sätzen bzw. Bemerkungen kann o.B.d.A. von einer L erzeugenden kontextfreien reduzierten Grammatik ohne ε-Regeln und ohne Kettenregeln ausgegangen werden. Für alle Regeln (u, v) dieser Grammatik gilt: $v \in M^n \cup A$ mit $n \geq 2$.
Die Regeln (u,v) mit $v = m_1 \ldots m_n$ für $n \geq 3$ werden ersetzt durch die neuen Regelmengen $R_{(u,\ v)} = \{(u,\ m_1[m_2 \ldots m_n]_{(u,\ v)}),\ \ldots\ ,([m_i \ldots m_n\ ]_{(u,\ v)},\ m_i[m_{i+1} \ldots m_n]_{(u,\ v)}),\ \ldots\ ,([m_{n-1} \ldots m_n]_{(u,\ v)},\ m_{n-1}m_n)\}$ mit den neuen Metazeichenmengen $M_{(u,\ v)}=\{[m_i \ldots m_n]_{(u,\ v)} \mid 2 \leq\ i \leq n-1\}$ und $M_{(u, v)} \cap M = \emptyset$.

Die so entstehende Grammatik ist kontextfrei, reduziert und in Chomsky-Normalform (die rechten Seiten der neuen Regeln sind Wörter aus $(M_{(u, v)} \cup M)^2$) und erzeugt L.

Insgesamt kann also eine reduzierte Chomsky-Normalform zu L konstruiert werden, wenn ausgehend von einer beliebigen L erzeugenden kontextfreien Grammatik zunächst eine äquivalente reduzierte, dann dazu eine äquivalente Grammatik ohne ε-Regeln und ohne Kettenregeln und schließlich die Grammatik nach obiger Konstruktion bestimmt wird.

Beispiel: Für die reduzierte kontextfreie Grammatik ohne ε-Regeln und ohne Kettenregeln mit der Regelmenge $S \to aT \mid bY,\ T \to aTT \mid bS \mid b,\ Y \to bYY \mid aS \mid a$ entsteht mit den Metazeichen $\underline{A}$, $\underline{B}$ für a, b die äquivalente Grammatik mit der Regelmenge $S \to \underline{A}T \mid \underline{B}Y,\ T \to \underline{A}TT \mid \underline{B}S \mid b,\ Y \to \underline{B}YY \mid \underline{A}S \mid a,\ \underline{A} \to a,\ \underline{B} \to b$. Mit den Metazeichen [TT], [YY] entsteht nach obiger Konstruktion die zur Ausgangsgrammatik äquivalente reduzierte Chomsky-Normalform mit der neuen Regelmenge $S \to \underline{A}T \mid \underline{B}Y,\ T \to \underline{A}[TT] \mid \underline{B}S \mid b,\ Y \to \underline{B}[YY] \mid \underline{A}S \mid a,\ [TT] \to TT,\ [YY] \to YY,\ \underline{A} \to a,\ \underline{B} \to b$.
Als Ableitungsbaum für das Wort aabbab bzgl. dieser Chomsky-Normalform erhalten wir einen im Inneren binären Wurzelbaum W mit $|N| \leq 3 \cdot 2^{\text{Höhe}(W)-1} - 1$, $\text{Höhe}(W) > \text{ld}(\,l\,(\text{Rand}(W)))$ und $l\,(\text{Rand}(W)) \leq 2^{\text{Höhe}(W)-1}$.



Definition**:** <u>Greibach-Normalform</u>

> Eine kontextfreie Grammatik G = (M, A, R, S) heißt in <u>Greibach-Normalform</u>, wenn für alle Regeln $(u, v) \in R$ gilt: $u \in M$ und $v \in AM^*$.

Satz: Existenz einer erzeugenden Greibach-Normalform

Zu jeder kontextfreien Sprache L mit $\varepsilon \notin L$ gibt es eine sie erzeugende kontextfreie Grammatik in Greibach-Normalform.

Beweis:

1) Ersetzt man in einer kontextfreien Grammatik die Regeln (m, pm'q) mit $(m', v_i) \in R$ durch die Regeln (m, pv_iq), dann erhält man eine äquivalente kontextfreie Grammatik.
2) Nach Einführung neuer Metazeichen $\overline{m}$ können die Regeln (m, mq_i) mit $(m, v_i) \in R$ und $v_i \neq mq_i$ durch die Regeln $(m, v_i \overline{m})$, $(\overline{m}, q_i)$, $(\overline{m}, q_i \overline{m})$ ersetzt werden.
3) O.B.d.A. kann davon ausgegangen werden, daß L durch die kontextfreie Chomsky-Normalform G=(M, A, R, S) mit $M=\{m_1, \ldots, m_l\}$ erzeugt wird.

Nach der ersten Bemerkung kann diese Grammatik durch sukzessives Ersetzen von Regeln äquivalent so umgeformt werden, daß für die Regeln (m_i, m_jq) gilt $j \leq i$. Weitere Umformung nach Bemerkung 2) führt zu einer äquivalenten Grammatik, deren Regeln (m, v) die Bedingung $v \neq wq$ erfüllen. Die Regeln der so entstehenden Grammatik haben die Form (m_i, m_jm'), (m_i, a), (m_i', v'), (m_i, b) mit $a, b \in A$, $m_i, m_j \in M$, m', m_i' sind neue Metazeichen, l(v)=2 und j>i. Durch rückwärtige Ersetzungen können die Regheln (m_i, m_jm') bzw. (m_i', v') durch (m_i, aq_i) bzw. (m_i', a_iq_i) mit $a_i \in A$ und $q_i \neq \varepsilon$ substituiert werden. Die damit entstehende kontextfreie Grammatik G'=(M', A, R', S) ist äquivalent zu G und enthält nur Regeln (u, v) mit $v \in AM'^2 \cup AM' \cup A$ (Spezialfall der Greibach-Normalform).

Bemerkung: Wegen der speziellen Form der Regeln sind Greibach-Normalformen auch für die Anwendung im Zusammenhang mit Implementierungen von Interesse.

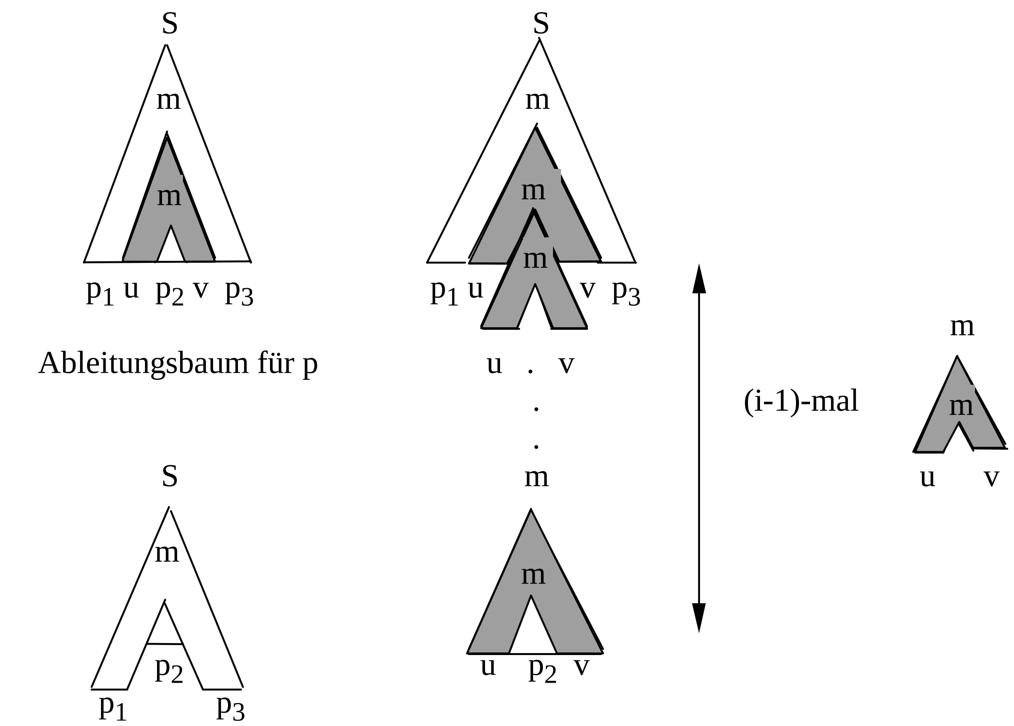
In Analogie zu den regulären Sprachen, gilt eine Pumping-Eigenschaft nach

Satz: Pumping-Lemma für kontextfreie Sprachen

Zu jeder kontextfreien Sprache L existiert eine nat. Zahl $n_L \geq 0$, so daß für alle $p \in L$ mit $l(p) \geq n_L$ gilt: Es gibt eine Zerlegung $p = p_1up_2vp_3$ mit $l(up_2v) \leq n_L$, $l(uv) \geq 1$ und für alle $i \geq 0$ sind die Wörter $p_1u^ip_2v^ip_3 \in L$.

Beweis: Für $L \neq \emptyset$ können wir nach früher bewiesenen Eigenschaften eine reduzierte Chomsky-Normalform angeben, die L \{ε} erzeugt. Für jedes durch eine solche Grammatik erzeugte Wort p gilt $l(p) \leq 2^{n-1}$, wo n die maximale Länge der Pfade im Ableitungsbaum für p ist. Wenn demnach $n_L=2^{|M|}$ (M Metazeichenmenge der erzeugenden Grammatik) und $l(p) \geq n_L$, dann muß es im Ableitungsbaum für p einen Pfad der Länge /M/+1 geben. Alle inneren Knoten (nicht Blatt) dieses Pfades sind mit Metazeichen bewertet. Der Ableitungsbaum enthält also mindestens /M/+1 Metazeichen, d.h., mindestens ein Metazeichen m kommt zweimal vor. Betrachten wir den kleinsten Unterbaum der m als Wurzel und als inneren Knoten besitzt. Dieser hat als Randbewertung das Wort up_2v mit der Länge $\geq n_L$. Das Wort uv kann nicht das leere Wort sein, da wir den kleinsten Unterbaum ausgewält hatten.

Wie aus nachfolgender Skizze zu ersehen ist, müssen auch die Bäume mit den Randbewertungen $p_1p_2p_3$ und $p_1u^ip_2v^ip_3$ für $i \geq 1$ Ableitungsbäume bzgl. der gewählten Grammatik sein. Die Zahl n_L kann also so gewählt werden, daß die Ableitungsbäume mit der Randbreite (Anzahl der Blätter = Länge der erzeugten Wörter aus L) $\geq n_L$ mindestens die Höhe /M/+1 erreichen.



Ableitungsbaum für $p_1p_2p_3$ Ableitungsbaum für $p_1u^ip_2v^ip_3$

Beispiel: Die Sprache $L=\{a^nb^nc^n \mid n\geq 1\}$ ist <u>nicht</u> kontextfrei. Es müßte sonst nach Pumping-Lemma ein $n_L>0$ und für das Wort $p=a^{n_L}b^{n_L}c^{n_L}$ eine Zerlegung $p=p_1up_2vp_3$ mit $l(up_2v)\leq n_L$, $l(uv)>1$ geben, wofür $q=p_1p_2p_3\in L$ (i=0) ist. Daraus würde folgen, daß $up_2v\in a^*b^* \cup b^*a^*$ und $l_a(uv)\neq l_b(uv)$ oder $l_a(uv)\neq l_c(uv)$ oder $l_b(uv)\neq l_c(uv)$ gilt. ($l_x(p)$ = Anzahl der mit x besetzten Stellen in p.)
Damit wäre aber auch $l_a(q)\neq l_b(q)$ oder $l_a(q)\neq l_c(q)$ oder $l_b(q)\neq l_c(q)$, d.h., $q\notin L$, denn in allen Wörtern aus L kommen a, b, c jeweils gleich oft vor. L kann demnach nicht kontextfrei sein.

Folgerung: Allgemeiner haben wir damit nachgewiesen, daß keine der nicht endlichen Teilmengen von
$\{p\in \{a,b,c\}^* \mid l_a(p)=l_b(p)=l_c(p)\}$ eine kontextfreie Sprache ist.

Definition: Die Funktionen f von A in 2^{B^*} mit $f(a)\subseteq B^*$ werden <u>(A,B)-Substitutionen</u> genannt und auf Wörter $p\in A^*$ und Sprachen $L\subseteq A^*$ wie üblich erweitert zu

$$f(\varepsilon)=\{\varepsilon\},\ f(pa)=f(p)\,f(a) \text{ und } f(L)=\bigcup_{p\in L} f(p).$$

Im Spezialfall $/f(a)/=1$ für alle $a\in A$ wirkt f als Homomorphismus von A^* nach B^*.

<u>Satz</u>**:** <u>Substitutionsgeschlossenheit</u>
Wenn L kontextfrei und für jedes $a\in A$ auch f(a) kontextfrei, dann ist auch f(L) kontextfrei.

Beweis: Seien G=(M, A, R, S) bzw. $G_a=(M_a, B, R_a, S_a)$ mit disjunkten Mengen M, M_a kontextfreie Grammatiken, die die Sprachen L bzw. f(a) erzeugen. Die kontextfreie Grammatik G'=(M', B, R', S) mit $M'=M \cup A \cup \bigcup_{a\in A} M_a$ und $R'=R \cup \{(a, S_a) \mid a\in A\} \cup \bigcup_{a\in A} R_a$ erzeugt dann offenbar die Sprache f(L).

Folgerung: Die kontextfreien Sprachen sind abgeschlossen gegenüber Homomorphismen.

***Satz*: Abgeschlossenheitseigenschaften**
a) Wenn L_i kontextfrei, dann sind auch $L_1 \cup L_2$, $L_1 \circ L_2$ und L_0^* kontextfrei.
b) Die kontextfreien Sprachen sind nicht abgeschlossen gegenüber Durchschnitt- und Komplementbildung.

Beweis:
a) Wenn $L_i=L(G_i)$ mit $G_i=(M_i, A, R_i, S_i)$ und $M_1 \cap M_2 = \emptyset$, dann ist $L_1 \cup L_2 = L(G_\cup)$,
$L_1 \circ L_2 = L(G_\circ)$ bzw. $L_0^* = L(G_*)$ mit
$G_U = (M_1 \cup M_2 \cup \{S\}, A, R_1 \cup R_2 \cup \{(S, S_1), (S, S_2)\}, S)$,
$G_\circ = (M_1 \cup M_2 \cup \{S\}, A, R_1 \cup R_2 \cup \{(S, S_1 S_2)\}, S)$,
$G_* = (M_0 \cup \{S\}, A, R_0 \cup \{(S,S_0 S), (S,\varepsilon)\},S)$
b) Die kontextfreien Grammatiken G: $S \to YT$, $T \to cT \mid c$, $Y \to aYb \mid ab$
bzw. G': $S \to Y'T'$, $T' \to bT'c \mid bc$, $Y' \to aY' \mid a$ erzeugen die kontextfreien Sprachen
$L(G) = \{a^m b^m c^n \mid m, n \geq 1\}$ bzw. $L(G')=\{a^l b^r c^r \mid l, r \geq 1 \}$.
Der Durchschnitt $L(G) \cap L(G') = \{a^i b^i c^i \mid i \geq 1\}$ ist aber nicht kontextfrei und wegen
$\overline{L_1 \cup L_2} = \overline{L_1} \cap \overline{L_2}$ kann damit auch die Abgeschlossenheit gegenüber Komplement nicht vorliegen.

2.4 Kontextabhängige Sprachen

Bei den Typ-1 Grammatiken hatten wir zunächst zwischen monotonen und kontextabhängigen Grammatiken unterschieden. Die kontextabhängigen Grammatiken sind nach Definition auch monoton. Umgekehrt kann zu jeder monotonen Grammatik auch eine äquivalente kontextabhängige Grammatik angegeben werden. Bevor wir dies zeigen, führen wir zunächst typunabhängig die separierten Grammatiken ein.

Definition: Separierte Grammatik
Eine Grammatik G = (M, A, R, S) heißt separiert, wenn für jede Regel $(u, v) \in R$ gilt: $u, v \in M^+$ oder $u \in M$ und $v \in A \cup \{\varepsilon\}$.

Satz: Zu jeder Grammatik G kann eine äquivalente separierte Grammatik G_S konstruiert werden.

Beweisskizze: Für die Zeichen a aus A werden Doppelgänger $a' \in A'$ gebildet und zu M als neue Metazeichen hinzugefügt. R´ bezeichne die Regelmenge, die aus R entsteht, wenn die Zeichen a durch ihre Doppelgänger a´ ersetzt werden. Zu der so entstehenden Regelmenge werden die Regeln (a´, a) für alle $a \in A$ hinzugefügt. Falls in R´ Regeln (u, ε) mit $u \notin M \cup A'$ auftreten, dann nehmen wir diese Regeln als weitere neue Metazeichen auf und ersetzen sie in der Regelmenge durch die Regeln $(u, (u, \varepsilon))$, $((u, \varepsilon), \varepsilon)$. Die so insgesamt entstehende Menge von Metazeichen bzw. Regeln bezeichnen wir durch M_S bzw. R_S . Die Grammatik $G_S = (M_S, A, R_S, S)$ ist separiert und erzeugt die Sprache L (G) , d.h., ist äquivalent zu G.

Satz: Zu jeder monotonen Grammatik G kann eine äquivalente kontextabhängige Grammatik G´ konstruiert werden.

Beweisskizze: O.B.d.A. kann zunächst G = (M, A, R, S) als separiert angenommen werden. Für die Regeln $r = (u, v) \in R$ mit $u,v \in M^+$ führen wir neue Metazeichen a^r_i für $1 \le i \le |u|$ und neue kontextabhängige Regeln
$(\alpha^r_1 \ldots \alpha^r_{i-1} u_i \ldots u_{l(u)} , \alpha^r_1 \ldots \alpha^r_i u_{i+1} \ldots u_{l(u)})$ für $1 \le i \le |u|$, (α^r_i , v_i) für $1 \le i \le \mathrm{Min}(|u|, |v|)$ und $(\alpha^r_{|u|} , v')$ mit $v = v_u v'$, $|u| = |v_u|$.
Bezeichnet M´ die so aus M gebildete neue Metazeichenmenge und R´ die durch diese Regelersetzung aus R entstehende neue Regelmenge, dann ist die Grammatik G´= (M´, A, R´, S) kontextabhängig und äquivalent zu G.
(Die Ableitungen nach G´ folgen denen nach G , denn die neuen kontextabhängigen Regeln sind an die Regelanwendung in G gebunden.)

Definition: Kuroda-Normalform
a) $Ord(G)=Max\{|v| \mid (u, v) \in R\}$ heißt die Ordnung der Grammatik G=(M,A,R,S).
b) G heißt längentreu , falls für alle $(u, v) \in R$ gilt: u = S oder |u| = |v| und s nicht in u .
c) G heißt Kuroda-Normalform, falls Ord (G) = 2 , G längentreu und mit $(S, xy) \in R$ ist x=S.

Satz: Zu jeder monotonen Grammatik existiert eine äquivalente Kuroda-Normalform

Beweisidee: Zunächst kann man die Ordnung einer monotone Grammatik G mit Ord (G) ≥ 3 stets dadurch um 1 reduzieren, wenn die Regeln (u, v) mit $|v| \geq 3$ durch neue regelgebundene Metazeichen über Zwischenschritte monoton realisiert werden. Diese Ordnungsreduktion wird wiederholt angewendet bis Ord (G) = 2. Die Bedingung der Längentreue wird dadurch erreicht, daß die Regeln (x, y_1y_2) nach Einführung eines Metazeichens λ durch $(\lambda x, y_1y_2)$ ersetzt werden. Sichert man noch, daß das Zeichen λ jede Stelle erreichen kann und benutzt ein neues Startzeichen σ, dann erhält man eine Kuroda-Normalform, die durch die homomorphe Substitution $f(\sigma) = S$, $f(\lambda) = \varepsilon$, $f(x) = x$ in eine zu G äquivalente Grammatik transformiert wird.

Im weiteren können wir dann voraussetzen, daß eine monotone Grammatik in Kuroda-Normalform vorliegt.

Beispiel: Wir betrachten die Sprache $L = \{a^nb^na^n \mid n \geq 1\}$. Diese nach Pumping-Lemma nicht kontextfreie Sprache wird von der monotonen Grammatik G = ({S, A, a, b}, {a, b},R, S) mit R = {(S, aba), (S, aSAa), (aA, Aa), (bA, bb)} erzeugt.
(Die erste Regel erzeugt das Wort für n=1 bzw. wird als Abschlußregel zur Beseitigung von S angewendet. Durch wiederholte Anwendung der zweiten Regel entstehen zunächst Wörter der Form $a^kS(Aa)^k$ und nach Anwendung der ersten Regel die Wörter $a^{k+1}b(Aa)^k$.
Daraus kann das Metazeichen A nur durch k-malige Anwendung der vierten Regel beseitigt werden, wobei mit Hilfe der dritten Regel die Reihenfolge von A und a vertauschbar ist. Insgesamt entstehen somit die Wörter $a^{k+1}b^{k+1}a^{k+1}$ mit $k \geq 1$. Da die Anwendung der Reihenfolge zwangsläufig ist, können auch keine anderen Wörter erzeugt werden.)
Als äquivalente separierte Grammatik entsteht nach oben skizziertem Verfahren die Regelmenge $\{(S, \alpha\beta\alpha)$., $(S, \alpha SA\alpha)$, $(\alpha A, A\alpha)$, $(\beta A, \beta\beta)$, (α, a), $(\beta, b)\}$ mit den neuen Metazeichen α und β für a und b. Um eine äquivalente kontextabhängige Grammatik zu erhalten, ist die Regel $(\alpha A, A\alpha)$ zu ersetzen durch die Regeln (α, δ), $(\delta A, \delta\eta)$, (δ, A), (η, α).
Um eine äquivalente Kuroda-Normalform zu erhalten, sind in der separierten Grammatik die Regel $(S, \alpha\beta\alpha)$ zu ersetzen durch die Regeln $(\lambda S, \mu\alpha)$, $(\lambda\mu, \alpha\beta)$ und die Regel $(S, \alpha SA\alpha)$ zu ersetzen durch die Regeln $(\lambda S, \pi\alpha)$, $(\lambda\pi, \alpha\psi)$, $(\lambda\psi, SA)$. Das neue Startzeichen σ wird mit der Regel (σ, S) in das alte Startzeichen S überführt. Mit der Regel $(\lambda x, x\lambda)$ für $(x \in M)$ kann das Zeichen λ an eine beliebige Stelle gebracht werden.

Für kontextabhängige Sprachen (Typ-1) gelten weiter folgende Eigenschaften:

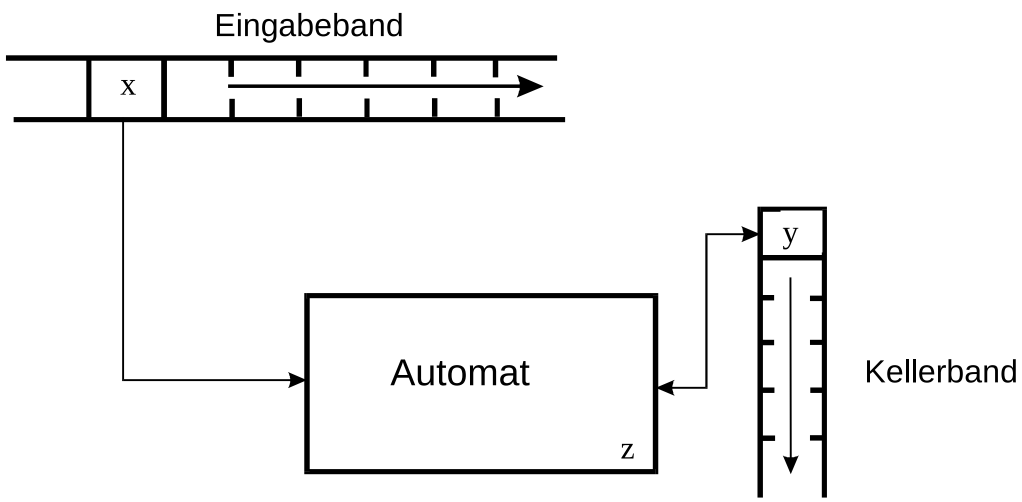
Satz: a) Jede kontextabhängige Sprache ist entscheidbar.
b) Jede kontextfreie Sprache, die nicht das leere Wort enthält, ist auch kontext-abhängig.
c) Die Klasse der kontextabhängigen Sprachen ist abgeschlossen gegen über Vereinigung, Komplementbildung, Verkettung, Durchschnittsbildung, inversen Homomorphismen, positiver Hüllenbildung (L^+).
d) Für kontextabhängige Sprachen L ist $L = \emptyset$ und $L = X^+$ unentscheidbar.

Beweisidee zu a) Man betrachte den endlichen Graphen, der die Wörter der Länge $\leq n$ als Knoten und die Regelanwendungen aus der Grammatik als Kanten hat. Eine Wort p mit $|p| \leq n$ gehört genau dann zur Sprache, wenn es eine Ableitung vom Startzeichen (Wurzelknoten) nach p als Pfad in diesem Graphen gibt. Diese Pfade können effektiv aufgesucht werden.

Die Typ-1 Sprachen bilden also eine echte Teilmenge der Typ-0 (aufzählbaren) Sprachen und auch der entscheidbaren Sprachen. Die Typ-1 Sprachen umfassen die Typ-2 (kontextfreien) Sprachen bis auf die Sprachen, die das leere Wort enthalten.

3.1 Kellerautomaten und kontextfreie Sprachen

Zur Beschreibung formaler Sprachen wurden bisher Regelsysteme verwendet, die formale Sprachen als Wortmengen erzeugen (generieren). Wir betrachten jetzt Verfahren, mit denen entschieden werden kann, ob ein gegebenes Wort zu einer bestimmten Sprache gehört. Wir sprechen dann im Gegensatz zur Regelgrammatik, die die Sprache generiert, von einer *akzeptiven* Grammatik bzw. von einem *Akzeptor*. Wie schon bei endlichen Automaten im Zusammenhang mit regulären Sprachen geben wir das Verfahren als abstrakte Maschine an.
Dazu erweitern wir zunächst den Begiff des endlichen Automaten, indem wir neben dem Eingabespeicher (ROM) einen Lese-Schreib-Speicher (RAM) zur Verfügung stellen, diesen einschränkend aber wie einen Keller benutzen. Dieser Speicher kann wieder Wörter über einem (Keller-) Alphabet aufnehmen und über die Operationen *push* (Einschreiben in den Keller), *pop* (Löschen des obersten Kellerelements) und *top* (Lesen des obersten Kellerelementes) benutzt werden. Wenn p, q Wörter über dem Kelleralphabet Y und y Element von Y, dann gilt für diese Operationen *push*(p, q) = pq, *pop*(py) = p und *top*(py) = y. Dabei steht das oberste Kellerelement am rechten Ende des Kellerwortes, d.h. wir lesen den Keller von rechts nach links. (Natürlich kann das auch umgekehrt festgelegt werden.) Die so festgelegten abstrakten Maschinen (siehe Schema) werden Kellerautomaten genannt.



Definition: <u>Kellerautomat</u>

Eine Struktur K = (X, Y, Z, h, z_0, S, F) heißt (endlicher) <u>Kellerautomat</u>, wenn

a) X (Eingabealphabet),Y (Kelleralphabet), Z (Zustandsmenge) nichtleere endliche Mengen,

b) $z_0 \in Z$ (Anfangszustand), $S \in Y$ (Startsymbol),
$F \subseteq Z$ (Endzustandsmenge),

c) h eine Funktion aus $(X \cup \{\varepsilon\}) \times Z \times Y$ in die Menge aller endlichen Teilmengen von $Z \times Y^*$.

Bemerkung: Der Kellerautomat ist allgemein nicht-deterministisch. Die Elemente der endlichen Mengen h(x, z, y), wo x das gelesene Zeichen des Eingabebandes oder das leere Wort, z den vorliegenden Zustand und y das oberste Kellersymbol bezeichnet, sind Paare (z, q) aus Folgezustand z und zu schreibendem Kellerwort q als mögliche Reaktionen des Automaten. Falls dabei immer nur höchstens eine Reaktion möglich ist, heißt K deterministisch .

Arbeitsweise des Kellerautomaten K:

Die momentane Situation von K wird durch eine Konfiguration $k = (p, z, q)$ beschrieben, wo p das gegebene Eingabewort (Belegung des Eingabebandes ab aktueller Position), z den aktuellen Automatenzustand und q das vorliegende Kellerwort (Belegung des Kellerbandes) angibt.
Die Konfigurationen (p, z_o, S) heißen Anfangskonfigurationen.
Die Konfigurationen $(\varepsilon, z, \varepsilon)$ bzw. (ε, z_f, q) mit $z_f \in F$ werden als Endkonfigurationen bezeichnet.
Mit Hilfe der Funktion h wird zu jeder Konfiguration k eine Menge von Folgekonfigurationen k' (in Zeichen: $k \vdash k'$) wie folgt bestimmt:

$(p, z, q) \vdash (p', z', q')$ genau dann, wenn $p=xp'$, $q=q_1y$, $q'=q_1q_1'$ und $(z', q_1') \in h(x, z, y)$

für $y \in Y$, $q_1, q_1' \in Y^*$ und $x \in (X \cup \{\varepsilon\})$.

Durch $\vdash^*$ bezeichnen wir wieder die transitive und reflexive Hülle von $\vdash$ und beschreiben damit Konfigurationsfolgen. Dabei ist $k \vdash^* k'$ genau dann, wenn $k = k'$ oder es gibt eine Folge $k_o ,..., k_n$ von Konfigurationen mit $k_o = k$, $k_n = k'$ und $k_i \vdash k_{i+1}$ für alle $0 \leq i < n$.

Definition: Akzeptierte Sprache

a) Die Wortmenge $L_\varepsilon(K) = \{p \mid p \in X^* \text{ und es existiert ein } z \in Z \text{ mit } (p, z_o, S) \vdash^* (\varepsilon, z, \varepsilon) \}$

heißt die vom Kellerautomaten K mit leerem Keller akzeptierte Sprache .

b) Die Wortmenge $L_F(K) = \{ p \mid p \in X^* \text{ und } (p, z_o, S) \vdash^* (\varepsilon, z_f, q) \text{ für ein } z_f \in F \text{ und } q \in Y^* \}$

heißt die vom Kellerautomaten K mit Endzustand akzeptierte Sprache.

Bemerkung: Allgemein ist $L_\varepsilon (K) \neq L_F (K)$. Wie später gezeigt wird, kann man durch Modifikation des Kellerautomaten K den einen in den anderen Fall überführen.

Beispiel: Der Kellerautomat K = ({ 0,1 }, { a,b,c }, (z_0, z_1 }, h, z_0, a, ∅) mit h nach Tabelle

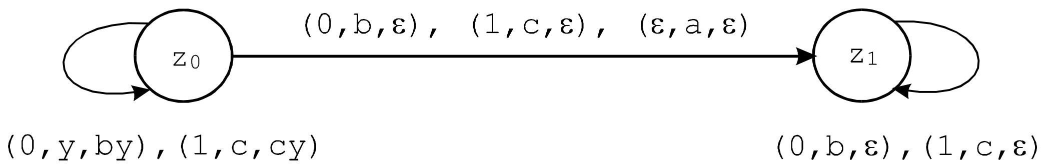
h	z_0			z_1		
a	(z_0, ba)	(z_0, ca)	(z_1, ε)	-	-	(z_1, ε)
b	(z_0, bb)(z_1, ε)	(z_0, cb)	-	(z_1, ε)	-	-
c	(z_0, bc)	(z_0, cc)(z_1, ε)	-	-	(z_1, ε)	-
x	0	1	ε	0	1	ε

akzeptiert z.B. das Wort 001100 durch die Konfigurationenfolge
(001100, z_0, a), (01100, z_0, ba), (1100, z_0, bba), (100, z_0, bbca),
(100, z_1, bbc), (00, z_1, bb), (0, z_1, b), (ε, z_1, ε) bzw.
das Wort 00 durch die Konfigurationenfolge
(00, z_0, a), (0, z_0, ba), (0, z_1, b), (ε, z_1, ε) bzw.
das Wort ε durch die Konfigurationenfolge (ε, z_0, a), (ε, z_1, ε).

Von diesem Kellerautomaten wird mit leerem Keller die Sprache L_ε (K) = { $p\ \tilde{p}$ | p ∈ {0,1}* } akzeptiert. An den nichtdeterministischen Stellen rät der Automat mit dem Übergang zum Zustand z_1 die Mitte des Eingabewortes. Diese Sprache kann von keinem deterministischen Kellerautomaten akzeptiert werden.

<u>Bemerkung:</u> Kellerautomaten können auch wieder durch Graphen beschrieben werden, wobei den Zuständen die Knoten und den durch h mit (z´, q) ∈ h (x, z, y) möglichen Übergängen die mit (x, y, q) beschrifteten Kanten entsprechen.

Der Kellerautomat aus dem obigen Beispiel ist durch den folgenden Graphen bestimmt.



Die Übergänge von z nach z´ für (z´, q) ∈ h (ε, y, z) heißen <u>ε-Übergänge</u>.

Satz: Akzeption mit leerem Keller und Akzeption mit Endzustand

Zu jedem Kellerautomaten K gibt es einen Kellerautomaten K´ mit L_ε (K) = L_F (K´) bzw. L_F (K) = L_ε (K´) .

Beweis: 1) L_ε (K) = L_F (K´) :

(K akzeptiert mit leerem Keller, K´ akzeptiert mit Endzustand)

Der Kellerautomat K = (X, Y, Z, h, z_0, S, ∅) wird durch den Kellerautomat K' = (X, Y´, Z´, h´, z_0´ , S´, {z_ε}) simuliert.

Wir führen ein neues Startsymbol S´ ein und erweitern die Zustandsmenge Z durch Hinzunahme eines Anfangszustandes z_0´ und eines Endzustandes z_ε zu der neuen Zustandsmenge Z´.

$S' \notin Y$, $z_0', z_\varepsilon \notin Z$, $Y' = Y \cup \{S'\}$, $Z' = Z \cup \{ z_0', z_\varepsilon \}$

Für alle x ∈ X, y ∈ Y, z ∈ Z ist h' und h identisch. Lediglich für das neue Startsymbol und den Anfangszustand z_0´ konstruieren wir h´ derart, daß ausgehend vom Startzustand z_0´ und vom Startsymbol S´ in den Zustand z_0 mit der Kellerinschrift S´S übergegeangen wird. Die neu entstandene Konfiguration ist die Startkonfiguration des Kellerautomaten K. Akzeptiert der Automat K mit leerem Keller, so ergibt sich die Kellerinschrift von K' zu S´ε. Anschließend wird in den Endzustand z_ε übergegangen.

$h'(\varepsilon, z_0', S') = \{ (z_0, S'S) \}$,
$h'(\varepsilon, z, S') = \{ (z_\varepsilon, \varepsilon) \}$,
$h'(x, z, y) = h(x, z, y)$ für alle $x \in X, y \in Y, z \in Z$.

Falls K determiniert ist, dann ist auch K´ nach dieser Konstruktion determiniert.

2) L_F (K) = L_ε(K´)

(K akzeptiert mit Endzustand, K´ akzeptiert mit leerem Keller)

Der Kelleautomat K = (X, Y, Z, h, z_0, S, Z_F) wird durch den Kellerautomat K´= (X, Y´, Z´, z_0´, h´, S´, ∅) simuliert.

Wir konstruieren eine neue Zustandsmenge Z´ durch Hinzunahme der Zustände z_0´,z_ε´. Das Kelleralphabet Y´ ergibt sich aus dem Kelleralphabet Y, erweitert um ein neues Startsymbol S´.

$S' \notin Y$, $z_0', z_\varepsilon' \notin Z$, $Y' = Y \cup \{S'\}$, $Z' = Z \cup \{ z_0', z_\varepsilon' \}$ und

Die Überführungsfunktion h´ wird so gebildet, daß bei erreichen eines Endzustandes (z ∈ Z_F) in einen neuen Zustand z_ε´ übergegangen wird, um anschließend den Keller vollständig zu leeren. Insgesamt gilt:

$h'(\varepsilon, z_0', S') = \{ (z_0, S'S) \}$,
$h'(\varepsilon, z_\varepsilon', y) = \{ (z_\varepsilon', \varepsilon) \}$ für alle $y \in Y'$,
$h'(\varepsilon, z, y) = \{ (z_\varepsilon', \varepsilon) \}$ für alle $y \in Y, z \in Z_F$,
$h'(x, z, y) = h(x, z, y)$ für alle $x \in X, y \in Y, z \in Z$.

Satz: Kellerautomaten und kontextfreie Sprachen

Eine Sprache L ist genau dann kontextfrei, wenn es einen Kellerautomat K mit $L = L_\varepsilon(K)$ gibt.

Beweis: a) Wenn L kontextfrei, dann gibt es einen Kellerautomaten K mit $L_\varepsilon(K) = L$. O.B.d.A. können wir voraussetzen, daß L durch eine Grammatik $G = (M, A, R, S)$ in reduzierter Chomsky-Normalform erzeugt wird, d.h., $L = L(G)$.
Wir bilden den Kellerautomaten
$K = (A, (M \cup A), \{z\}, h, z, S, \emptyset)$ mit
$h(x, z, x) = \{(z, \varepsilon)\}$, $h(\varepsilon, z, u) = \{(z, v)\}$ für $x \in A$, $u \in M$ und $(u, v) \in R$.

Man kann zeigen, daß es zu jeder Linksableitung eines Wortes p aus L bzgl. G eine Konfigurationsfolge von K gibt, so daß $S \Rightarrow p$ gdw. $(p, z, S) \vdash^* (\varepsilon, z, \varepsilon)$.
Der Kellerautomat simuliert mit den Übergängen (ε, u, v) die Linksableitung und schreibt jeweils das Spiegelbild der durch die Regelanwendung erzeugten Wörter als Zwischenergebnisse in den Keller. Die Überführung (x, x, ε) dient dazu, den Keller von Eingabesymbolen aus A zu säubern, die bei der Simulation im Keller entstehen. Genauer wird der Beweis induktiv über die Länge der Ableitung bzw. Konfigurationsfolge geführt.

b) Wenn K ein Kellerautomat, dann ist die Sprache $L_\varepsilon(K)$ kontextfrei.
Zu $K = (X, Y, Z, h, z_o, S, \emptyset)$ wird unter der Annahme $\varepsilon \notin L_\varepsilon(K)$ die kontextfreie Grammatik $G = (M, X, R, S)$ konstruiert , wofür $M = X \cup Y \cup \{ [z, y, z'] \mid y \in Y, z, z' \in Z \}$ und $R = R_S \cup R_X$ mit
$R_S = \{ (S, [z_o, y, z]) \mid y \in Y, z \in Z \}$ und
$R_X = \{ ([z, y, z'] , x[z^n , y^n , z^{n-1}] [z^{n-1} , y^{n-1} , z^{n-2}] ... [z^1 , y^1 , z']) \mid$
$n \geq 0 , z, z', z^i \in Z, x \in X \cup \{\varepsilon\}$, und $(z^n , y^1 ... y^n) \in h(x, z, y)\}$.

Man kann zeigen, daß für die so gewählte Grammatik $[z_o, S, z'] \Rightarrow p \in X^*$ genau dann gilt, wenn $(p, z_o, S) \vdash^* (\varepsilon, z', \varepsilon)$. Da mit den Regeln aus R_S die Ableitung $S \Rightarrow [z_o, S, z']$ erzeugt werden kann, folgt schließlich $p \in L_\varepsilon(K)$ genau dann, wenn $S \Rightarrow p \in L(G)$. Für $\varepsilon \in L_\varepsilon(K)$ ist noch eine entsprechende Regel in G zu ergänzen.

Folgerung: Zu jedem Kellerautomaten kann ein Kellerautomat mit einem Zustand angegeben werden, der dieselbe Sprache akzeptiert. (Die dem Satz entsprechende Konstruktion dieses Kellerautomaten geht zu Lasten des Kellerumfangs.)

Definition: Deterministischer Kellerautomat, deterministisch kontextfreie Sprache

a) Ein Kellerautomat $K = (X, Y, Z, h, z_o, S, F)$ heißt deterministisch , falls zu jeder Konfiguration k höchstens eine Folgekonfiguration k´ mit $k \vdash k'$ existiert.
b) Eine formale Sprache L heißt deterministisch kontextfrei, falls ein deterministischer Kellerautomat K mit $L_F (K) = L$ existiert.

Satz: Durchschnitt mit regulären Sprachen

Wenn $L \subseteq X^*$ (deterministisch) kontextfrei und $R \subseteq X^*$ regulär, dann ist $L \cap R$ (deterministisch) kontextfrei .

Beweis: Zu den nach Voraussetzung existierenden (deterministischen) Kellerautomaten $K = (X, Y, Z, h, z_0, S, F)$ und endlichen Automaten $A = (X, Z´, f, z_0´, Z_F´)$ mit $L = L_F(K)$ und $R = L(A)$ konstruieren wir den (deterministischen) Kellerautomaten $K´= (X, Y, Z´´, h´, z_0´, S, F´´)$, wo $Z´´= Z \times Z´$, $z_0´´= [z_0, z_0´]$, $F´´= F \times Z_F´$ und $([\bar{z} ,\bar{z}´] , q) \in h´(x, y, [z, z´])$ falls $(\bar{z} , q) \in h (x, y, z)$ und $f(x, z´) = \bar{z}´$, $([\bar{z} ,\bar{z}´], q) \in h´(\varepsilon, y, [z, z´])$ falls $(\bar{z} , q) \in h (\varepsilon, y, z)$.

Dann kann man zeigen:

$L_{F´´}(K´) = L_F(K) \cap L (A)$, denn für $p \in X^*$, $q \in Y^*$, $\bar{z} \in F$, $\bar{z}´\in Z_F´$ ist $(p, [z_0, z_0´], S) \vdash (\varepsilon, [\bar{z} ,\bar{z}´], q)$ gdw. $(p, z_0, S) \vdash (\varepsilon,\bar{z} , q)$ und $f(p,z_0) = \bar{z}´$.

Satz: Abgeschlossenheit gegenüber inversen Homomorphismen

Wenn $L \subseteq X^*$ (deterministisch) kontextfrei und $f : \bar{X}^* \to X^*$ eine Substitution mit $f(pq) = f(p) f(q)$ für alle $p,q \in \bar{X}^*$ (Homomorphismus), dann ist $f^{-1}(L)$ (deterministisch) kontextfrei.

Beweis: Zu den nach Voraussetzung existierenden (deterministischen) Kellerautomaten $K = (X, Y, Z, h, z_0, S, F)$ mit $L = L_F(K)$ konstruieren wir den (deterministischen) Kellerautomaten $K´= (X, Y, Z´, h´, [z_0, \varepsilon] , S, F \times \{ \varepsilon \})$ mit $Z´= Z \times \{ p \mid \text{existiert } p´\in \bar{X}^* , \bar{x} \in \bar{X} \text{ mit } f(\bar{x}) = p´p \}$ und $h´(\bar{x}, y, [z, \varepsilon]) = \{ ([z, f(\bar{x})] , y) \mid \bar{x} \in \bar{X}, y \in Y \}$, d.h., zur Eingabe $\bar{x}$ wird $f(\bar{x})$ als Eingabe für K erzeugt, bzw. $h´(\varepsilon, y, [z, p] \} = \{ ([\bar{z} , p], q) \mid (\bar{z} , q) \in h (\varepsilon, y, z) \}$, d.h., es werden ε-Übergänge von K simuliert, bzw $h´(\varepsilon, y, [z, xp]) = \{ ([z, p], q) \mid \{ z, q) \in h (x, y, z) \}$,
d.h., es werden Übergänge von K zur Eingabe x simuliert.

Dann kann gezeigt werden, daß für $z_f \in F$ gilt:
$(p,[z_0,\varepsilon],S) \vdash^*_{K´}(\varepsilon,[z_f,\varepsilon],q)$ gdw. $(f(p),z_0,S) \vdash^*_K(\varepsilon, z_f, q)$, d.h. $L_F (K´)=f^{-1}(L)$.

Definition: Min(L)= {p | p∈L und für alle q: wenn q∈L Anfangsstück von p, dann ist p=q} als die Menge aller Wörter aus L, für die kein echtes Anfangswort zu L gehört, wird als *Minimum* von L bezeichnet.

Satz: Abgeschlossenheit gegenüber Minimumbildung
Wenn L deterministisch kontextfrei, dann ist auch Min(L) deterministisch kontextfrei.

Beweis: Zu dem nach Voraussetzung existierenden deterministischen Kellerautomaten $K = (X, Y, Z, h, z_0, S, F)$ mit $L_F (K) = L$ konstruieren wir den deterministischen Kellerautomaten $K'=(X,Y,Z,h',z_0,S,F)$ mit $h'=\{h \mid X\times Y\times Z \setminus F\}$, d.h., aus h werden die von Endzuständen ausgehenden Übergänge gestrichen. Dann gilt: $L_F (K') = Min (L)$.

Bemerkung: Es existieren kontextfreie Sprachen, die nicht determininistisch kontextfrei sind.

Beispiel: Die Sprache $L = \{ p\tilde{p} \mid p \in \{ a, b \}^* \}$ wird durch die Grammatik mit den Regeln $S \to aSa \mid bSb \mid \varepsilon$ erzeugt, ist also kontextfrei. Wenn wir annehmen, daß L deterministisch kontextfrei ist, dann wären $L' = L \cap (ab)^+ (ba)^* (ab)^* (ba)^+ = \{(ab)^m (ba)^n (ab)^n (ba)^m \mid m \geq 1, n \geq 0 \}$ und $Min(L') = \{ p \mid p \in L'$ und $n < m\}$ auch deterministisch kontextfrei, woraus im Widerspruch zu früher die Kontextfreiheit von $f^{-1}(Min (L')) = \{ a^m b^n a^n b^m \mid 0 \leq n < m \}$ mit der homomorphen Substitution $f (a) = ab$, $f (b) = ba$ folgen würde.

Folgerung: Nichtdeterministische Kellerautomaten sind hinsichtlich der von ihnen akzeptierten Sprachen ausdruckstärker als deterministische Kellerautomaten.

Bemerkung: Es existieren deterministische Kellerautomaten K mit nicht regulärem $L_\varepsilon(K)$.

Beispiel: Die nicht reguläre Sprache $L = \{ a^n b a^n \mid n \geq 0 \}$ wird von dem deterministischen Kellerautomaten $K = (\{ a, b \}, \{S\}, \{ z_0, z_1, z_2 \}, h, z_0, S)$ mit $h(a, z_0, S) = (z_0, SS)$, $h(b, z_0, S) = (z_1, \varepsilon)$, $h(a, z_1, S) = (z_1, \varepsilon)$, $h (b, z_1, S) = (z_2, S)$, $h(a, z_2, S) = (z_2, S)$, $h (b, z_2, S) = (z_2, S)$ zum leeren Keller im Zustand z_1 akzeptiert, d.h., $L = L_\varepsilon(K)$.

Folgerung: Deterministische Kellerautomaten sind hinsichtlich der von ihnen akzeptierten Sprachen ausdrucksstärker als endliche Automaten.

Satz: Eigenschaften deterministisch kontextfreier Sprachen

a) Wenn L deterministisch kontextfrei und R regulär, dann ist die Quotientensprache $L / R = \{ p \mid$ existiert $q \in R$ mit $pq \in L \}$ deterministisch kontextfrei.
b) Wenn $L \subseteq X^*$ deterministisch kontextfrei, dann ist auch ihr Komplement $X^* \setminus L$ deterministisch kontextfrei.
c) Zu jeder deterministisch kontextfreien Sprache L existiert ein deterministischer Kellerautomat K ohne ε-Übergänge für Endzustände mit $L = L_F (K)$.
d) Es gibt deterministische Kellerautomaten K, wo $L_\varepsilon(K)$ nicht deterministisch kontextfrei ist.
e) Deterministisch kontextfreie Sprachen sind nicht abgeschlossen unter Vereinigung, Verkettung, Iteration und Homomorphismen.

3.2 Turing-Automaten und Regel-Sprachen

Um die Klasse der von Automaten akzeptierten Sprachen über die kontextfreien Sprachen hinaus zu erweitern, heben wir die beim Kellerautomaten angenommene Beschränkung des RAM-Speichers als Keller wieder auf und betrachten ein unendliches Speicherband, wo an beliebigen Stellen (Zellen) gelesen und geschrieben werden kann. Wir benutzen dazu das nach beiden Seiten offene Eingabeband. Um beliebige Stellen dieses Bandes zu erreichen, geben wir dem Automaten die Möglichkeit, neben dem Lese- und Schreibvorgang, ausgehend von einer Zelle die linke oder rechte Nachbarzelle aufzusuchen. Zusammen mit dem zu schreibenden Symbol wird diese Bewegungsmöglichkeit durch eine Ausgabereaktion des Automaten bestimmt. Wir folgen dem von *Turing* (1936) vorgeschlagenen Modell des Turing-Automaten und untersuchen die von diesem akzeptierte Sprachklasse.

Definition: Turing-Automat

Eine Struktur $T = (X, Y, Z, h, z_o, F, \#)$ heißt Turing-Automat , falls

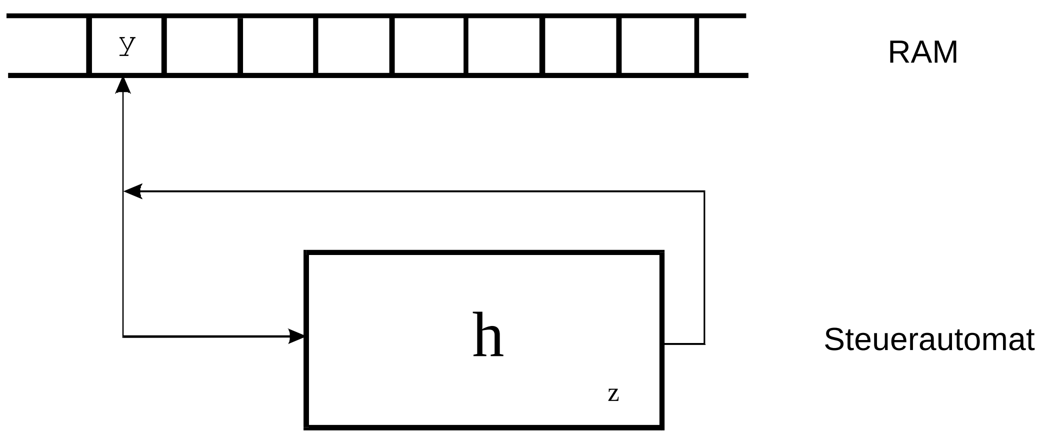
a) X, Y, Z nichtleere endliche Mengen, $X \subset Y$, $Y \cap Z = \emptyset$, $\# \in Y \setminus X$, $z_o \in Z$, $F \subseteq Z$ und

b) h eine Funktion aus $Y \times Z$ in die Potenzmenge über $Y \times Z \times \{ r, l, o \}$.
(Die im allgemeinen partielle Funktion h mit Tripelmengen als Funktionswerte kann auch als Relation über $Y \times Z \times Y \times Z \times \{ r, l, o \}$ aufgefaßt werden. r, l, o kennzeichnen die Bewegungsmöglichkeiten als Ausgabereaktion.)

Bezeichnungen:

X	Eingabealphabet,	Y	Bandalphabet,
Z	Zustandsmenge,	F	Endzustandsmenge,
z_o	Anfangszustand,	$\#$	Leerfeldsymbol (Blank),
h	Überführungsfunktion.		

Schematisch kann man einen Turing-Automaten wie folgt darstellen:



<u>Bemerkung:</u> Hinsichtlich der Realisierung des Speichers und seiner Nutzung sind noch andere Modellvorstellungen entwickelt worden, z.B. mehrere Speicher mit verschiedenen Grundalphabeten, andere Adressierungs- bzw. Bewegungsmöglichkeiten, die im wesentlichen aber keine grundsätzliche Erweiterungen gegenüber dem obigen Modell darstellen.

Zur Beschreibung der Arbeitsweise eines Turing-Automaten bei vorgegebener Beschriftung des Speicherbandes führen wir <u>Konfigurationen</u> als Wörter der Form pzq aus Y*ZY* ein, wo z den aktuellen Zustand des Steuerautomaten, p die Beschriftung des Speichers links von der adressierten Zelle, q die Beschriftung des Speichers ab der adressierten Zelle nach rechts und das erste Zeichen von q den Inhalt der adressierten Zelle bestimmt. Der Automat arbeitet immer auf einem endlichen Abschnitt des Speichers und außerhalb dieses Abschnittes ist der Speicher mit dem Symbol # (Blank) beschriftet.
Alle Konfigurationen #*pzq#* werden mit pzq identifiziert, d.h., vor p bzw. hinter q dürfen beliebig viele # geschrieben werden.
Zu einer Konfiguration pzq werden <u>Folgekonfigurationen</u> p´z´q´ als Verhaltensreaktion des Automaten wie folgt definiert:

Wenn $(y, z', r) \in h(x, z)$ mit $q = xw$, dann ist $p' = py$ und $q' = w$ möglich.
Wenn $(y, z', l) \in h(x, z)$ mit $p = vx'$, $q = xw$, dann ist $p' = v$ und $q' = x'yw$ möglich.
Wenn $(y, z', o) \in h(x, z)$ mit $q = xw$, dann ist $p' = p$ und $q' = yw$ möglich.

Die Folgekonfigurationen entstehen durch Anwendung folgender Substitutionsregeln:

Bei Bewegung r durch die Regel $zx \to yz'$,
bei l durch die Regel $zx \to z'y$
und bei o durch die Regel $x'zx \to z'x'y$ für beliebige $x' \in Y$.

Fassen wir diese Regeln zur Regelmenge R_T zusammen, dann kann der Übergang von der Konfiguration pzq zu einer Folgekonfiguration p´z´q´ wie üblich mit $pzq \vdash p'z'q'$ durch Regelanwendung beschrieben werden, wobei die Symbole # vor p bzw. hinter q nach Bedarf zu ergänzen sind.
Bezeichnet $\vdash^*$ wieder die reflexiv, transitive Hülle von $\vdash$, dann ist die von dem Turing-Automaten <u>akzeptierte</u> <u>Sprache</u> durch die Wortmenge

$$L(T) = \{ q \mid q \in X^* \text{ und es existiert ein } z_f \in F \text{ mit } z_0q \vdash^* vz_fw \}$$

gegeben.
Ein Wort q , als Anfangsbeschriftung des Speicherbandes, wird vom Turing-Automat genau dann akzeptiert, wenn die Konfigurationsfolge zu einem Endzustand führt. Das Eingabewort q muß dabei nicht notwendig vollständig gelesen werden. Die Akzeption sagt nichts darüber aus, nach wieviel Schritten ein Endzustand erreicht wird. Da h allgemein partiell ist, muß nicht zu jeder Konfiguration eine Folgekonfiguration definiert sein. In diesem Fall sagt man , daß der Turing-Automat <u>hält</u>. Wird ein Wort nicht akzeptiert, dann kann die Konfigurationsfolge unter Umständen unbeschränkt fortsetzbar sein, ohne daß ein Endzustand angenommen wird oder der Automat anhält.

<u>Modellvarianten :</u>

a) Turing-Automaten mit halbseitig unendlichem Speicher

Da der Automat zu jedem Zeitpunkt nur einen endlichen Speicherbereich benutzt hat, kann bei notwendigen Bewegungen über das Speicherende vorher eine gegenläufige Verschiebung des benutzten Speicherbereiches vorgenommen werden. (Auffüllen mit Blanks.)

b) Mehrband-Turing-Automaten

Mehrere Speicherbänder können zu einem Speicherband mit mehreren Spuren zusammengefaßt werden. Die Zellen des gemeinsamen Speicherbandes werden entsprechend mit Tupel von Symbolen beschriftet, die auf den gegebenen Speicherbänder stehen. Zur Festlegung der jeweils aktuell zu lesenden Zelle wird das dort vorhandene Symbol in der Beschriftung des gemeinsamen Speicherbandes auf der entsprechenden Spur durch einen Doppelgänger ersetzt. Diese Doppelgänger sind als neue Symbole in die Menge Y aufzunehmen. Sobald die jeweilige Zelle nicht mehr aktuell ist, wird an dieser Stelle auf der entsprechenden Spur wieder das Originalsymbol eingetragen.

Beide Varianten können daher auf das ursprüngliche Modell des Turing-Automaten zurückgeführt werden.

Definition: Deterministischer Turing-Automat
Ein Turing-Automat $T = (X, Y, Z, h, z_0, F, \#)$ heißt deterministisch, falls h eine allgemein partielle Funktion aus $Y \times Z$ in $Y \times Z \times \{ r, l, o \}$ ist, d.h., $|\, h(y, z) \,| \leq 1$ für alle $y \in Y$ und $z \in Z$.

Satz: Äquivalenz nichtdeterministischer und deterministischer Turing-Automaten
Zu jedem nichtdeterministischen Turing-Automaten läßt sich ein deterministischer Turing-Automat angeben, der dieselbe Sprache akzeptiert.

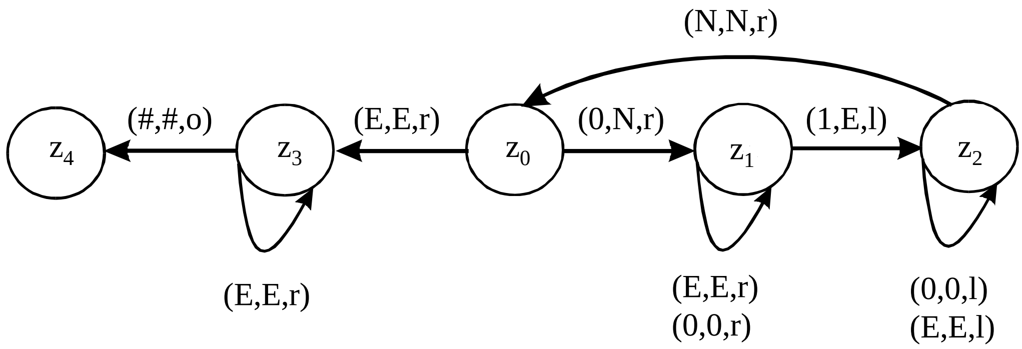
Beweisidee: O.B.d.A. kann von Turing-Automaten mit einem Speicherband und jeweils höchstens zwei möglichen Folgekonfigurationen ausgegangen werden. Die bildbaren Konfigurationsfolgen können dann als Wörter über einem binären Alphabet codiert werden. Es wird jetzt ein Dreiband-Automat konstruiert, der im ersten Band das Eingabewort und im zweiten Band die Kodewörter für die Konfigurationsfolgen des nichtdeterministischen Automaten speichert. Nachdem ein Kodewort gespeichert wurde, wird die entsprechende Konfigurationsfolge auf dem dritten Band deterministisch ausgeführt.

Definition: Aufzählbare (berechenbare) Sprache
Eine formale Sprache L heißt (rekursiv) aufzählbar, wenn es einen Turing-Automaten T mit $L = L(T)$ gibt, d.h., der L akzeptiert.

Beispiel: Der Turing-Automat $T=(\{0, 1\}, \{0, 1, N, E, \#\}, \{z_0, z_1, z_2, z_3, z_4\}, h, z_0, \{z_4\}, \#)$ mit h nach Tabelle

	0	1	N	E	#
z_0	(N, z_1, r)	-	-	(E, z_3, r)	-
z_1	$(0, z_1, r)$	(E, z_2, l)	-	(E, z_1, r)	-
z_2	$(0, z_2, l)$	-	(N, z_0, r)	(E, z_2, l)	-
z_3	-	-	-	(E, z_3, r)	$(\#, z_4, o)$
z_4	-	-	-	-	-

kann in Analogie zu den anderen Automatenbegriffen auch durch den Graphen

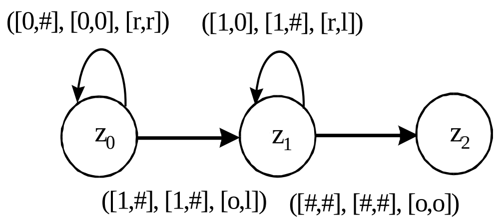


festgelegt werden.
Der Automat T akzeptiert die Sprache L (T) = { 0^n 1^n | n ≥ 1 }. Dabei bedeuten die Zustände:

- z_o (Anfangszustand) ersetzt eine rechtsstehende 0 durch N mit Rechtsbewegung.
- z_1 sucht nach rechts die erste 1 und ersetzt diese durch E mit Linksbewegung.
- z_2 sucht nach links das erste N und geht danach mit Rechtsbewegung in den Zustand z_o.
- z_3 sucht nach rechts das erste Blank und geht danach mit Rechtsbewegung in z_4 über.
- z_4 (Endzustand) ist der akzeptierende Zustand .

Zum Eingabewort 0011 entsteht die Konfigurationsfolge z_o0011 ├ Nz_1011 ├ $N0z_111$ ├ Nz_20E1 ├ z_2N0E1 ├ Nz_o0E1 ├ NNz_1E1 ├ $NNEz_11$ ├ NNz_2EE ├ Nz_2NEE ├ NNz_oEE ├ $NNEz_3E$ ├ $NNEEz_3$ ├ $NNEEz_4$.

Dieselbe Sprache wird durch den Zweiband-Automaten mit folgendem Graph akzeptiert.



Die Reaktionen auf den beiden Bändern sind jeweils in eckige Klammern gesetzt. Zunächst schreibt der Automat im Zustand z_o die Nullen vor der ersten 1 vom ersten (Eingabe-)Band auf das zweite Band. Dann wird im Zustand z_1 geprüft, ob danach auf dem ersten Band soviele Einsen stehen wie auf dem zweiten Band Nullen. Nur dann geht der Automat in den akzeptierenden Zustand z_2 über.

Satz: Turing-Automaten und Typ-0 - Sprachen

Eine formale Sprache L ist genau dann vom Typ-0 , wenn es einen sie akzeptierenden Turing-Automaten T mit L = L (T) gibt.

Beweis: a) Konstruktion einer Typ-0 Grammatik G zum Turing-Automaten T mit L (T) = L (G):

Sei T = (X, Y, Z, h, z_0, F, #) und $S \notin Y \cup Z$. Dann bilden wir $M = Y \cup Z \cup \{S\}$ und das zu T gehörige Regelsystem R_T , so daß $p \in L(T)$ genau dann, wenn gilt $z_0p \Rightarrow q_1z_fq_2$ mit $q_i \in Y^*$, $z_f \in F$ mit Hilfe der Regeln aus R_T (bis auf Blanks).

Wir bilden die Regelmenge $R = R_T \cup \{z_f \to S, yS \to S, Sy \to S \mid z_f \in F, y \in Y\}$. Dann gilt $p \in L(T)$ genau dann, wenn $z_0p \Rightarrow S$ mit den Regeln aus R. Die Grammatik G´ = (M, A, R´, S) mit $A = X \cup Z \cup \{\#\}$ und wo R´ aus R durch Umkehrung der Regeln entsteht erzeugt dann die Sprache L (G´) mit der Eigenschaft :

$\{ z_0p \mid p \in L(T) \} = L(G') \cap \{ z_0 \}A^*$. Aus G´ kann leicht die gesuchte Grammatik G mit L (G) = L (T) erzeugt werden.

b) Konstruktion eines Turing-Automaten T zur Typ-0 Grammatik G mit L(G)=L (T)

Idee: Man bilde einen Zweiband-Automat , der auf dem ersten Band das Eingabewort p aufnimmt und damit auf dem zweiten Band die Ableitungen $S \Rightarrow q$ der Grammatik G bildet. (Die Übergangsfunktion h des Turing-Automaten entspricht den Regeln der Grammatik, wie eingangs erläutert.) Danach werden Beschriftungen beider Bänder verglichen. Der Automat akzeptiert, wenn p = q .

Folgerungen:
1. Eine Sprache ist vom Typ-0 genau dann, wenn sie (rekursiv) aufzählbar ist.
2. Eine Sprache ist (rekursiv)aufzählbar genau dann, wenn sie von einem deterministischen Turing-Automaten (mit einem Band) akzeptiert wird.

Definition: Entscheidbare Sprachen

Eine formale Sprache $L \subseteq X^*$ heißt (rekursiv) entscheidbar, wenn es einen deterministischen Turing-Automaten T mit L = L (T) gibt, der für jede Eingabe $p \in X^*$ hält .

Bemerkung: Für solche Sprachen ist durch T ein Entscheidungsverfahren gegeben, mit dem für jedes Wort aus X* festgestellt werden kann, ob es zu L gehört oder nicht. Diese Entscheidung kann am Zustand, den T nach dem Anhalten einnimmt, abgelesen werden. In jedem Fall erreicht der Turing-Automat T eine Konfiguration, zu der keine Folgekonfiguration existiert.

Satz: <u>Zusammenhang zwischen aufzählbaren und entscheidbaren Sprachen</u>
Eine Sprache $L \subseteq X^*$ ist genau dann entscheidbar, wenn L und ihr Komplement $X^* \setminus L$ aufzählbar ist.

Beweisidee: a) Nach Definition ist $L \subseteq X^*$ genau dann entscheidbar, wenn ihr Komplement $X^* \setminus L$ entscheidbar ist. Jede entscheidbare Sprache ist aber auch aufzählbar.
b) Wenn L und $X^* \setminus L$ aufzählbar sind, dann existieren Turing-Automaten T und T´ mit $L(T) = L$ und $L(T') = X^* \setminus L$. Man bilde einen Turing-Automaten mit zwei Bändern, der auf dem ersten Band den Automaten T und auf dem zweiten Band den Automaten T´ simuliert bei dem gleichen Eingabewort $p \in X^*$. Die Zustände dieses Automaten sind die Zustandspaare der Automaten T und T´. Der Automat soll anhalten (keine Folgekonfiguration definiert), sobald bei einem der Automaten T oder T´ein Endzustand ereicht wird. Das Wort p wird akzeptiert, wenn der Automat auf Band eins anhält, d.h., wenn T anhält.

Satz: <u>Existenz nicht entscheidbarer Sprachen</u>
Es existieren (rekursiv) aufzählbare Sprachen, die nicht entscheidbar sind.

Beweisidee: Jede endliche algebraische Struktur kann über einem zweielementigen Alphabet {0, 1} kodiert werden, d.h., man kann eine injektive Funktion Φ angeben, die zu jeder Struktur eindeutig eine Folge (Kodewort) über {0, 1} bestimmt, aus der umgekehrt auch die kodierte Struktur wieder ermittelt werden kann. Die Werte von Φ und Φ^{-1} sind effektiv zu konstruieren. Entsprechendes gilt für Tupel aus (endlichen) Wörtern und Strukturen.Sei Φ eine solche Kodierungsfunktion der Paare (p, T), wo $p \in X^*$ und T Turing-Automat über X.
Wir führen die <u>(Universal)-Sprache</u> $L_u = \{ \Phi(p, T) \mid p \in L(T) \} \subseteq \{0, 1\}^*$ ein. L_u ist (rekursiv) aufzählbar und wird durch einen Zweiband-Automaten T_u akzeptiert, der auf dem ersten Band zur Eingabe $\Phi(p, T)$ entsprechend Φ^{-1} zunächst p und T bestimmt und danach auf dem zweiten Band den Automaten T bei der Eingabe p simuliert. T_u akzeptiert $\Phi(p, T)$, wenn p von T akzeptiert wird. T_u heißt <u>Universal-Automat</u> , da er beliebige T simuliert.
Andererseits ist die <u>(Diagonal)-Sprache</u> $L_d = \{ \Phi'(T) \mid \Phi'(T) \notin L(T) \} \subseteq \{0, 1\}^*$, wo $\Phi'(T)$ Kodewort von T, nicht aufzählbar, denn sonst müßte es einen Turing-Automaten T_d mit $L(T_d) = L_d$ geben, wofür $\Phi'(T_d) \in L_d = L(T_d)$ genau dann, wenn $\Phi'(T_d) \notin L(T_d)$ ist.
Wäre nun $\{0, 1\}^* \setminus L_u$ aufzählbar, dann könnten wir L_d dadurch aufzählen, daß wir durch einen Turing-Automaten für die Eingabe $\Phi'(T)$ das Kodewort $\Phi(\Phi'(T), T)$ erzeugen und akzeptieren, wenn dieses Wort nicht zu L_u gehört. Wenn aber $\{0, 1\}^* \setminus L_u$ nicht aufzählbar ist, dann kann die durch T_u aufzählbare Sprache L_u auch nicht entscheidbar sein.

Folgerung: Die aufzählbaren Sprachen sind nicht abgeschlossen gegen Komplementbildung.

Defintion: <u>Turing-berechenbare Funktion</u>

Sei T ein deterministischer Zweiband-Automat mit dem Alphabet X_1 für das erste (Eingabe)-Band und X_2 für das zweite (Ausgabe)-Band. f_T bezeichne die Funktion , die genau für diejenigen Wörter p über X_1 definiert ist, für die T mit p als Beschriftung des Eingabebandes anhält, und der Funktionswert $f_T(p)$ als Beschriftung des Ausgabebandes entsteht.
Eine Funktion f aus X_1* nach X_2* heißt <u>(Turing)-berechenbar</u>, wenn es einen deterministischen Turing-Automaten T mit $f_T = f$ gibt.

<u>Eigenschaften:</u> 1. Jede im intuitiven Sinne berechenbare Funktion ist Turing-berechenbar.
2. Es gibt Funktionen, die nicht Turing-berechenbar sind.

Definition: <u>Turing-erzeugbare Sprache</u>

Sei T ein nichtdeterministischer Mehrband-Automat mit einem (Ausgabe)-Band, auf dem nur von links nach rechts geschrieben werden kann. Mit ERZ(T) bezeichnen wir die Menge aller Inschriften dieses Ausgabebandes ohne Blanks, die entstehen, wenn T angesetzt auf leere Bänder anhält.
Eine Sprache L heißt <u>Turing-erzeugbar</u>, wenn es einen nichtdeterministischen Turing-Automaten T mit ERZ(T) = L gibt.

<u>Eigenschaften:</u> 1. Jede Turing-erzeugbare Sprache ist (rekursiv)aufzählbar und umgekehrt.
2. Wenn L aufzählbar, dann gibt es einen Turing-Automaten der jedes Wort von L genau einmal erzeugt.
3. L ist genau dann entscheidbar, wenn es einen Turing-Automaten gibt, der die Wörter von L in aufsteigender Länge und genau einmal erzeugt.

Definition: <u>(rekursiv) reduzierbare Sprachen</u>

Eine Sprache $L_1 \subseteq X_1$* heißt auf die Sprache $L_2 \subseteq X_2$* (rekursiv) reduzierbar ($L_1 \leadsto L_2$), wenn es eine überall auf X_1* definierte Turing-berechenbare Funktion f nach X_2* gibt, für die $f(p) \in L_2$ genau dann gilt, wenn $p \in L_1$.

<u>Eigenschaften:</u> 1. Die Relation $\leadsto$ ist reflexiv und transitiv.
2. $L_1 \leadsto L_2$ genau dann, wenn $X_1^* \setminus L_1 \leadsto X_2^* \setminus L_2$.
3. Wenn $L_1 \leadsto L_2$ gilt und L_2 ist aufzählbar, dann ist auch L_1 aufzählbar.
4. L ist genau dann aufzählbar, wenn $L \leadsto L_u$ mit L_u (Universal)-Sprache.

Definition: *Eigenschaften* P aufzählbarer Sprachen sind Teilklassen der Klasse L_o der Typ-0 Sprachen, d.h., Prädikate über L_o. Eine Eigenschaft P heißt nicht-trivial, wenn P mindestens eine und nicht alle Sprachen aus L_o umfaßt, d.h., es gilt: $\emptyset \neq P \subset L_o$.

Satz: Existenz nicht entscheidbarer Eigenschaften aufzählbarer Sprachen (Satz von Rice)
Jede nicht-triviale Eigenschaft aufzählbarer Sprachen ist unentscheidbar.

Beweisidee: L sei eine aufzählbare Sprache mit L = L (T) und P sei eine nicht-triviale Eigenschaft aufzählbarer Sprachen , die auf L aber nicht auf $\emptyset$ zutrifft, d.h., $L \in P$ und $\emptyset \notin P$. Eine solche Eigenschaft P existiert, da P entscheidbar genau dann, wenn (nicht P) entscheidbar ist.
Zu einem Wort q über dem Eingabealphabet von T konstruieren wir einen Turing-Automaten T_q der bei einem Wort p als Beschriftung des (Eingabe)Bandes genau dann anhält , wenn der Automat T bei p und der Automat T_u bei q anhält, d.h., es gilt :

$$L(T_q) = \{ p \mid p \in L \text{ und } q \in L(T_u) = L_u \}.$$

Also ist $L(T_q) = L$, falls $q \in L_u$ und $L(T_q) = \emptyset$, falls $q \notin L_u$.
Das bedeutet : $L(T_q) \in P$ genau dann, wenn $q \in L_u$.
Sei $L_p = \{\Phi'(T) \mid \Phi'(T)$ Kodewort des Turing-Automaten T und $L(T) \in P\}$. Wäre L_p entscheidbar, dann wäre auch entscheidbar, ob $L(T_q) \in P$ gilt oder nicht. Damit wäre aber im Widerspruch zu früher L_u entscheidbar. Also kann L_p nicht entscheidbar sein. Damit ist auch die nicht-triviale Eigenschaft P nicht entscheidbar.

Folgerung: Es ist nicht entscheidbar, ob eine beliebige (rekursiv) aufzählbare (Typ-0) Sprache leer, bzw. endlich, bzw. regulär, bzw. kontextfrei, bzw. kontextabhängig, bzw. entscheidbar ist. (Nicht-triviale Eigenschaften)

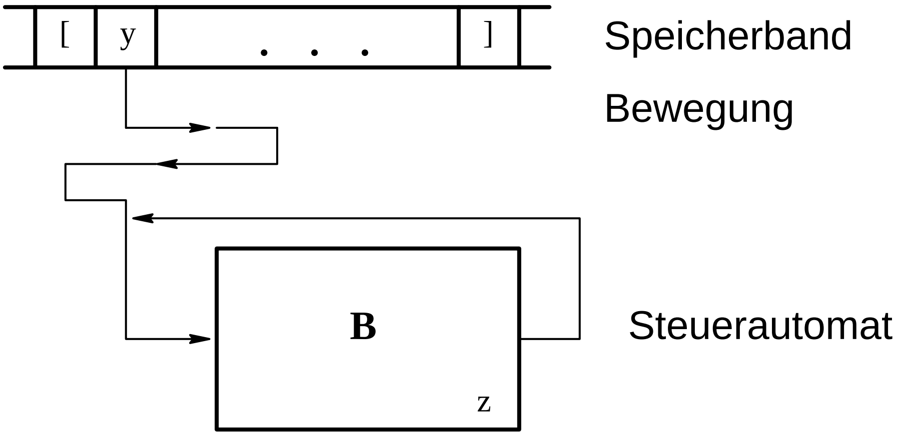
Bemerkungen:
1. Folgende Eigenschaften (rekursiv) aufzählbarer Sprachen L sind nicht (rekursiv) aufzählbar: $L = \emptyset$; $L = X^*$; L rekursiv-aufzählbar ; L nicht rekursiv-aufzählbar ; L regulär ; $L \setminus L_u \neq \emptyset$.
2. Folgende Eigenschaften (rekursiv) aufzählbarer Sprachen sind (rekursiv) aufzählbar : $L \neq \emptyset$; $|L| \geq k > 0$; $p \in L$ für ein vorgegebenes festes $p \in X^*$; $L \cap L_u \neq \emptyset$.
3. Es ist nicht entscheidbar, ob ein (beliebiger) Turing-Automat bei leerem Eingabeband (nur Blanks) hält. (Halteproblem: Verallgemeinerung des Satzes von Rice)

3.3 Linear-beschränkte Automaten und kontextabhängige Klassen

Die Monotonie der Typ-1 Grammatiken erlaubt es, das Speicherband des akzeptierenden Turing-Automaten auf den Bereich des gegebenen Eingabewortes linear zu beschränken.

Definition: Linear beschränkter Automat
Eine Struktur B = (X, Y, Z, h, z_o, F, [,]) heißt linear-beschränkter Automat, wenn
a) (X, Y, Z, h, zo, F) ein (nichtdeterministischer) Turing-Automat (ohne Blank),
b) [,] $\in$ Y \ (X $\cup$ Z) (linke bzw. rechte Randmarke) und
c) aus (y´, z´, o) $\in$ h (y, z) folgt y´ = [bzw.] genau dann, wenn y = [bzw.] und aus y = [bzw] folgt o $\neq$ l bzw. r . (Die Randmarken dürfen weder geschrieben, verändert, noch überschritten werden.)

Durch L (B) = { p | p $\in$ X* und [z_op] $\vdash^*$ [uz_fv] mit $z_f \in$ F } wird die von B akzeptierte Sprache bezeichnet.
Der Automat B kann nur auf dem durch das Eingabewort beschriebenen Bandbereich arbeiten.
Die Arbeitsweise von B wird durch das folgende Bild verdeutlicht:



Satz: Typ-1 Sprachen und linear-beschränkte Automaten

Eine Sprache L ist genau dann vom Typ-1, wenn es einen linear beschränkten Automaten B mit L (B) \ {ε} = L gibt.

(Das leere Wort ε kann durch einen Automaten B akzeptiert , durch eine monotone Grammatik aber nicht generiert (erzeugt) werden.)

Beweisskizze: a) Zu jeder monotonen Grammatik G = (M, A, R, S) existiert ein linear-beschränkter Automat B mit L (G) = L (B).
O.B.d.A. kann G als Kuroda-Normalform vorausgesetzt werden, wobei für die Regeln (u, v) aus R gilt: |u| = |v| $\leq$ 2 und u $\in$ M $\cup$ M^2 und S nicht in v oder u = S und v = Sy . Für jedes p $\in$ L (G) gibt es dann eine Ableitung S $\Rightarrow$ Sq $\Rightarrow$ p mit |q| = |p| -1 , d.h., es wird zunächst ein mit S beginnendes Wort der Länge von p aufgebaut. (Es ist nur ein Metazeichen m mit q = $m^{|p|-1}$ erforderlich.)

Man kann nun einen linear-beschränkten Automaten $B = (A, Y, Z, h, z_o, F, [,])$ angeben, wo h die längentreuen Regeln rückwärts ausführt und das eingegebene Wort dann akzeptiert wird, wenn zwischen den Randmarken ein Wort entsteht, daß mit den Regeln $(S, v) \in R$ aus S ableitbar ist.
(Man kann auch einen Automaten mit einem zweispurigen Band konstruieren, der auf der zweiten Spur nichtdeterministisch ein Wort aus L (G) erzeugt und dieses dann mit dem auf der ersten Spur stehenden Eingabewort vergleicht. Der Automat akzeptiert, wenn beide Spuren dasselbe Wort enthalten.)

b) Zu jedem linear-beschränkten Automaten B existiert eine monotone Grammatik G mit $L(B) \setminus \{\varepsilon\} = L(G)$.
Die Konstruktion von G wird analog zum entsprechenden Satz für Turing-Automaten durchgeführt, nur ist hier zu beachten, daß störende Hilfszeichen wegen Monotonie nicht gelöscht werden können. Mit den Metazeichen z^x, $z^[$, $z^]$, S, σ, ω für $x \in X$ und $z \in Z$ werden deshalb folgende Regeln konstruiert:

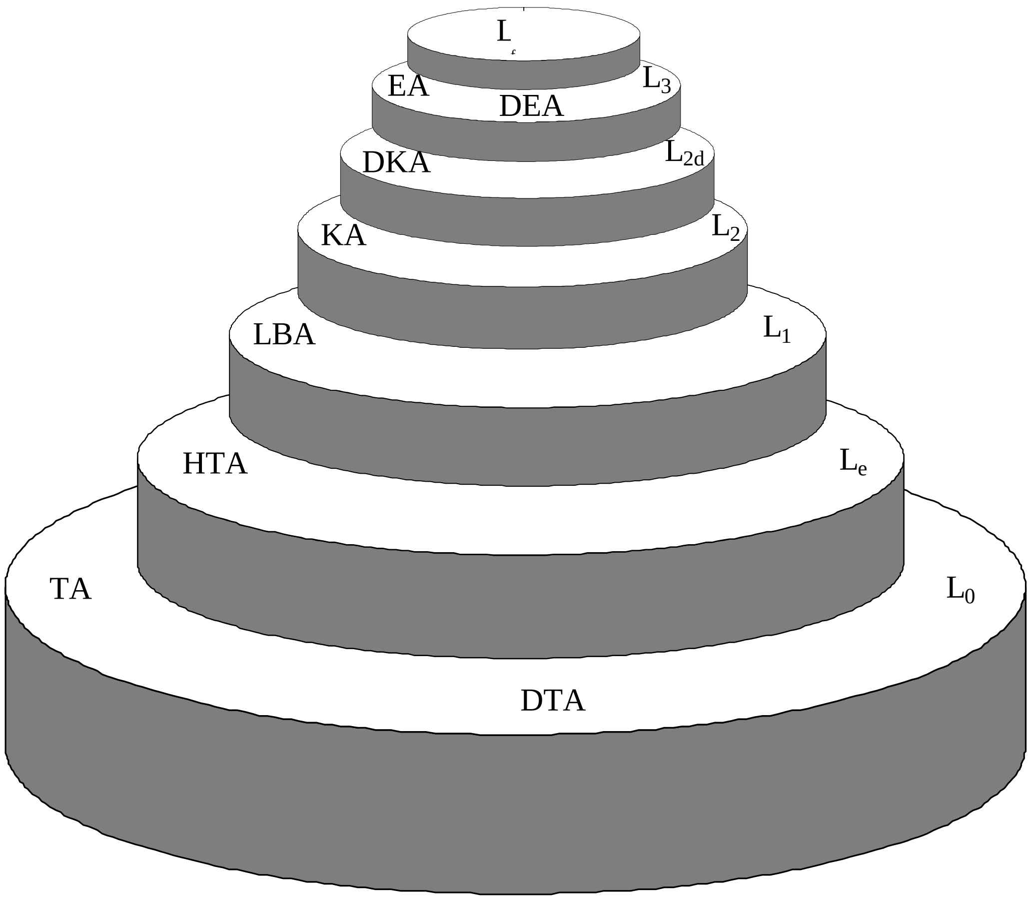
Ableitungsanfang: $(S, \sigma\omega)$, $(\sigma, x\sigma)$, (σ, x), $(\omega, z^])$ für $x \in X$ und $(], z_f, r) \in h(], z)$, $z_f \in F$.

Simulation von B: $(z'^{x'}, z^x)$ bei $(x', z', o) \in h(x, z)$,
$(x' z'^{x''}, z^x x')$ bei $(x', z', r) \in h(x, z)$, $x' \in X$,
$(z'^{x''} x', x' z^x)$ bei $(x', z', l) \in h(x, z)$, $x' \in X$ und
$(x' z'^], z^x)$, $([z^x, [x)$ bei $(x', z', r) \in h(x, z)$, $x \in X$.

Die Grammatik mit den so eingeführten Metazeichen und Regeln ist monoton und generiert die von ε verschiedenen Wörter aus L (B).

Die linear-beschränkten Automaten charakterisieren demnach gerade die Klasse der Typ-1 Sprachen.

3.4 Sprach- und Automatenklassen



L_f	endliche Sprachen	L_e	entscheidbare Sprachen
L_3	reguläre Sprachen (Typ-3)	EA DEA	endliche Automaten determ. endliche Automaten
L_{2d}	determ. kontextfreie Sprachen	DKA	determ. Kellerautomaten
L_2	kontextfreie Sprachen (Typ-2)	KA	Kellerautomaten
L_1	kontextabhänige Sprachen (Typ-1)	LBA	linear-beschränkte Automaten
L_0	aufzählbare Sprachen (Typ-0)	TA DTA HTA	Turing-Automaten determ. Turing-Automaten haltender Turing-Automat

Stichwortverzeichnis