# Mitschrift
# Formale Sprachen und Automaten

Prof. Braun
Sommersemester 2002
LaTeXed by Leo Bär

# Inhaltsverzeichnis

## Vorwort

Diese Mitschrift basiert auf meinen Unterlagen und Kopien der Unterlagen meiner Kommilitonen. Es kann daher keinerlei Garantie auf die Korrektheit der Aussagen in diesem Skript gegeben werden. Teilweise habe ich mir herausgenommen bestimmte Textstellen zu verändern, um sie für mich verständlicher zu machen. Auch die Gliederung wurde teilweise verändert.

Desweiteren ist die Darstellung von terminal und nonterminal Zeichen nicht einheitlich. Oft existieren auch noch Schreibfehler, welche zu Verwechslungen führen können (z.B. $w$ statt $\omega$). Wer solche findet bzw. Unklarheiten aufdeckt, kann mir das unter `Leo.Baer@gmx.de` mitteilen. Aber auch für Konstruktive Kritik bin ich sehr dankbar.

Leo Bär 12. Juni 2002

# Kapitel 1

# Varianten endlicher Automaten

### Deterministische Automaten

Oder auch DFA (deterministic finite automaton). Sei $M$ ein Automat, dann gilt $M = (V, \Sigma, \delta, q_0, F)$ mit

- $Q$ Zustandsmenge
- $\Sigma$ Alphabet
- $\delta$ Übergangsfunktion
- $q_0$ Anfangszustand
- $F$ Endzustände

Weiterhin gelte:

$$\delta : Q \times \Sigma \to Q \qquad \hat{\delta} : Q \times \Sigma^* \to Q$$
$$\hat{\delta}(q, \epsilon) := q \qquad \hat{\delta}(q, a) := \delta(q, a) \qquad \hat{\delta}(q, wa) := \hat{\delta}\big(\hat{\delta}(q, w), a\big)$$

Der Sprachschatz ist

$$T(M) := \{w \in \Sigma^* \big| \hat{\delta}(q_0, w) \in F\} = \bigcup_{q \in F} [w]_q \quad \text{mit} \quad [w]_q := \{x \in \Sigma^* \big| \hat{\delta}(q_0, x) = \hat{\delta}(q_0, w) = q\}$$

$T(M)$ ist eine reguläre Menge und eine Typ 3 Sprache

### Nichtdeterministische Automaten

Es gilt

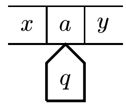
$$\delta(q, a) \subseteq Q \qquad \hat{\delta}(q, wa) = \bigcup_{r \in \hat{\delta}(q, w)} \hat{\delta}(r, a) \qquad T(M) = \{w \in \Sigma^* \big| \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$$

## 1.1 Der 2-Wege-Automat

Ein 2-Wege-Automat ist ein DFA mit Lesekopfbewegung in beide Richtungen.

**Definition 1.1 (2-Wege-Automat)** *Ein* ***2-Wege-Automat (2DFA)*** *ist ein DFA mit erweiterter Übergangsfunktion*

$$\delta : Q \times \Sigma \to Q \times \{R, L\}$$




*Also einem Übergang in einen neuen Zustand und einer Bewegung.*
*Notation:* $q\ a\ q'\{R|L\}$ *Konfiguration:* $xqay$.
*Seien* $\alpha, \beta$ *Konfigurationen, dann ist* $\alpha \vdash \beta$ *ein Übergang.*
*Weiterhin gilt*

$$T(M) = \left\{ w \in \Sigma^* \middle| q_o w \vdash^* wp, p \in F \right\}$$

*Ein Band wird nicht akzeptiert, falls:*

1. *Der rechte Rand nicht in einem Endzustand erreicht wird.*
2. *Der rechte Rand nicht erreicht wird.*
3. *Eine Bewegung über den linken Rand hinaus stattfinden.*

Dies sollte an einem kleinen Beispiel klarer werden.

**Beispiel**

Sei $\Sigma : 0\{0, 1\}$. Der 2DFA soll genau die Strings akzeptieren, die nicht 2 oder mehr aufeinander folgende 1er enthalten. Also:

$$Q = F = \{q_0, q_1\} \text{ mit } \qquad q_0 0 q_0 R \qquad q_0 1 q_1 R$$
$$q_1 0 q_0 R \qquad q_1 1 q_1 L$$

Damit

| | | | |
|---|---|---|---|
| $q_0\,101001 \vdash$ | $1\,q_1\,01001 \vdash$ | $10\,q_0\,1001 \vdash$ | $101\,q_1\,001 \vdash$ |
| $1010\,q_0\,01 \vdash$ | $10100\,q_0\,1 \vdash$ | $101001\,q_1 \vdash$ | akzeptiert |

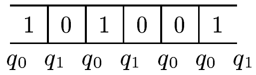
Aber

| | | |
|---|---|---|
| $q_0\,101101 \vdash$ | $1\,q_1\,01101 \vdash$ | $10\,q_0\,1101 \vdash$ |
| $101\,q_1\,101 \vdash$ | $10\,q_1\,1101 \vdash$ | $1\,q_1\,01101$ |

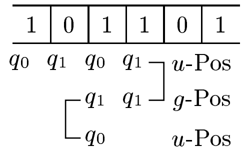
Hier sind der 2. und 6. Zustand identisch, es hat sich also eine unendliche Schleife gebildet.

---

Im Beispiel traf zum einen die Situation auf, dass der Automat im Zustand $q$ von links kam und $a$ auf dem Band stand. Der vorherige Zustand war also $\ldots qR$. Später traf der Automat im Zustand $q$ noch einmal auf $a$, kam aber von rechts (vorheriger Schritt $\ldots qL$). Es ergaben sich also folgende Situationen:




und




**Definition 1.2 (Crossing Sequence (CS))** *Die einer Grenze zugeordnete Folge von Zuständen heißt* ***crossing sequence***

**Lemma 1.3 (Eigenschaften von CS)** *Sei* $c := [q_1, q_2, \ldots, q_k]$ *eine CS. Dann gilt:*

- *Jedes* $q_i$ *auf ungerader Position (=:u-Pos.) wird von links her erreicht, jedes* $q_i$ *auf gerader Position (=:g-Pos.) wird von rechts her erreicht.*

- *Falls $w \in R(M)$, haben alle CS ungerade Länge*
- *Eine CS mit demselben $q$ auf 2 u-Positionen oder auf g-Positionen ist eine unendliche Schleife.*

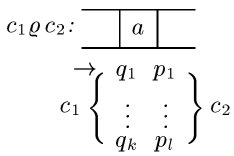
**Definition 1.4 (gültige CS)** *Eine CS c heißt genau dann* ***gültig****, wenn c von ungerader Länge ist und in c nie 2 u-Positionen identisch sind.*

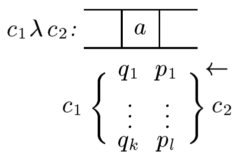
**Lemma 1.5** *Sei $s := |Q|$, dann besitzt jede gültige CS eine Länge $l \leq 2s - 1$*

**Beweis**
Sei $l > 2s - 1$. Da $l$ ungerade (gültige CS) gilt $l \geq 2s + 1$.
Sei $q \in Q$ auf $u$-Pos. $2s + 1$. Dann existieren links von $q$ noch $s$ $u$-Pos.en (1,3,4,...,2s-1).
Dann befinden sich aber auf diesen Position entweder zweimal der selbe Zustand oder lauter verschiedene Zustände. ist entwe-
Im 1. Fall wäre diese Teil CS ungültig, im 2. Fall ist $q$ unter den Zuständen und die gesamte der oder
CS ungültig. ■ hier ok?

**Definition 1.6 (Verträglichkeit)** *Seien $c_1 := [q_1, \ldots, q_k]$ und $c_2 := [p_1, \ldots, p_l]$ die unteren Enden zweier aufeinander folgender CS links/rechts von a. Dann heißt $c_1$ genau dann* ***linksverträglich*** *mit $c_2$ (Notation: $c_1 \varrho c_2$) über a, wenn $q_1 \in c_1$ von links her erreicht wird.*

*Entsprechend heißt $c_1$* ***rechtsverträglich*** *mit $c_2$ ($c_1 \lambda c_2$) über a genau dann, wenn $p_1 \in c_2$ von rechts her erreicht wird. Desweiteren soll gelten: $\epsilon \varrho \epsilon$ und $\epsilon \lambda \epsilon$*




**Lemma 1.7** *Sei $c_1 := [q_1, \ldots, q_k]$ und $c_2 := [p_1, \ldots, p_l]$, dann gilt:*

1.
$$c_1 \varrho c_2 \text{ (über } a) \Longleftrightarrow \big(q_1 a q_2 L \wedge ([q_3, \ldots, q_k] \varrho [p_1, \ldots, p_l])\big) \vee \big(q_1 a p_1 R \wedge ([q_2, \ldots, q_k] \lambda [p_2, \ldots, p_l])\big)$$

2.
$$c_1 \lambda c_2 \text{ (über } a) \Longleftrightarrow \big(p_1 a p_2 R \wedge ([q_1, \ldots, q_k] \lambda [p_3, \ldots, p_l])\big) \vee \big(p_1 a q_1 L \wedge ([q_2, \ldots, q_k] \varrho [p_2, \ldots, p_l])\big)$$

**Lemma 1.8**

1. *$\epsilon \varrho c$ ist nicht möglich für $c \neq \epsilon$.*
2. *$c \varrho \epsilon$ möglich für $c \neq \epsilon$, denn: $c = [q_1, \ldots], q_1 a q_2 L, [q_3, \ldots, q_k] \varrho \epsilon$.*
3. *$\epsilon \varrho c$ möglich für $c \neq \epsilon$, denn: $c = [p_1, p_2, p_3, \ldots], p_1 a p_2 R, \epsilon \lambda [p_3, \ldots, p_l]$.*
4. *$c \lambda \epsilon$ nicht möglich für $c \neq \epsilon$.*

**Satz 1.9** *Für jeden 2DFA $M$ existiert ein NFA $M'$ mit $T(M) = T(M')$.*

**Beweis**
**Idee:** Zustände des $M'$ sind die gültigen cs des $M$. Übergang $(c_1, a) \underset{M'}{\vdash} c_2$ bei $c_1 \varrho c_2$ über $a$ im 2DFA $M$.

1. $M' = \big(Q', \Sigma, \delta', q_0', F'\big)$, wobei gilt:
   $Q'$ ist die Menge aller gültigen cs des $M$
   $q_0' := [q_0]$ $\quad F' := \big\{[p] \big| p \in F\big\}$ $\quad \delta'(c,a) = \big\{d \in Q' \big| c\varrho d \text{ über } a\big\}$
2. Zeige $w \in T(M) \Longrightarrow w \in T(M')$.
   Sei $w = a_1 a_2 \ldots a_n$, $c_0, c_1, \ldots, c_n$ mit $c_0 = [q_O]$ und $c_n = [p]$.
   Zu zeigen also $c_{i-1}\varrho c$ über $a_i$ und $c_{i-1} = [q_1, \ldots, q_k], c_i = [p_1, \ldots, p_l]$. Dann ist der Zugang zu $q_1$ von links $\to q_1$, danach entweder $q_1 a_i q_2 L$ oder $q_1 a_i p_1 R$.
   Bleibt als für $c_{i-1}\varrho c_i$ zu zeigen, dass entweder
   $[q_3 \ldots q_k]\varrho[p_1 \ldots p_l]$ über $a_i$ oder
   $[q_2 \ldots q_k]\lambda[p_2 \ldots p_l]$ über $a_i$.
   Echt kürzer, also Fortsetzung bis $\epsilon\varrho\epsilon$ liefert $c_{i-1}\varrho c_i$ über $a_i$.
3. Zeige $w \in T(M') \Longrightarrow w \in T(M)$
   Sei $w = a_1 \ldots a_n \in T(M')$, dann $\underset{[=q_0]}{c_0} \overset{a_1}{\vdash} c_1 \overset{a_2}{\vdash} \cdots \vdash c_{n-1} \overset{a_n}{\vdash} \underset{=[p]}{c_n}$ mit $c_{i-1}\varrho c_i$ über $a_i$.
   Sei $w_i := a_1 \ldots a_i$; $\quad w_0 := \epsilon$; $\quad c_o \overset{a_1 \ldots a_i}{\underset{M'}{\vdash}} c_i$ und $c_i = [p_1 \ldots p_l]$ mit $l$ ungerade.
   Dann Zeige
   (a) Bei verarbeiten von $w_i$ durch $M$ ab $q_0$ verlässt $M$ das $a_i$ zum ersten Mal nach rechts im Zustand $p_1$.
   (b) Für $j \in \{2, 4, \ldots, l_1\}$ gilt, $M$ liest $a_i$ in $p_j$. Also $M$ verlässt $a_i$ nach rechts zum ersten Mal in $p_{j+1}$.

   **Induktionsanfang**
   Sei $i = 0$, $w_i = w_0 = \epsilon$, $c_0 = [q_0]$.
   Es gilt stets $[q_0] \overset{\epsilon}{\underset{M'}{\vdash}} [q_0]$, also (3a) erfüllt und (3b) leer.

   **Induktionsvorraussetzung**
   (3a) und (3b) gelten für $i_1$. Genauer:
   $c_0 \overset{a_i \ldots a_{i-1}}{\underset{M'}{\vdash}} c_{i-1}$, $c_{i-1} = [q_1, \ldots, q_k]$ mit $k$ ungerade.
   Dann gilt
   (a) $M$ verlässt $a_{i-1}$ zum ersten Mal nach rechts in $q_1$, also $qa_{i-1}q_1R$.
   (b) Für $j \in \{2, 4, \ldots, k-1\}$ gilt, $M$ liest $a_i$ in $q_i$ (kommt also von rechts).
   $M$ verlässt $a_{i-1}$ also zum ersten Mal nach rechts in $q_{j+1}$ mit $\tilde{q}a_{i-1}q_{j+1}R$.

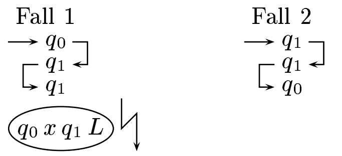
   **Induktionsschluss** $(i-1 \to i)$
   Sei $c_{i-1} = [q_1, \ldots, q_k]$, $c_i = [p_1, \ldots, p_l]$ und $c_{i-1}\varrho c_i$ über $a_i$. Zu (3a).
   $a_i$ erstesmal gelesen beim ersten Verlassen von $a_1 \ldots a_{i-1}$ nach rechts.
   Nach IV $a_i$ erstmalig gelesen in $q_1$. Nach Regel $q_1 a_i \ldots$ also $c_{i-1}\varrho c_i$ über $a_i$.
   In $M$ also entweder (Fall 1) $q_1 a_i p_1 R$ oder (Fall 2) $q_1 a_i q_2 L[q_3 \ldots q_k]\varrho[p_1 \ldots p_l]$
   Bei Fall 1 fertig.
   Bei Fall 2 kehrt $M$ bei $a_i$ in $q_2$ nach links um. $M$ liest $a_{i-1}$ also in $q_2$. Nach IV verlässt $M$ danach $a_{i-1}$ erstmalig nach rechts in $q_3$. Es gilt also dasselbe wie zuvor, aber mit echt verkürztem $c_{i_1}' = [q_3 \ldots q_k]$. Bei Fortsetzung tritt also entweder Fall 1 ein oder weitere Verkürzungen des $c_{i-1}$. Spätestens bei der Verkürzung des $c_{i-1}$ auf $[q_k]$ entsteht Fall 1.

   Der Beweis zu (3b) funktioniert analog.

∎

**Beispiel**

Sei $q_0 0 q_0 R$, $q_0 1 q_1 R$, $q_1 0 q_0 R$, $q_1 1 q_1 L$ mit $s = 2$
Also existieren $(2s - 1 = 3)$ Längen 1, 3 für gültige cs. Position 2 ist aber nur mit $L$ erreichbar. Auf Position 2 kann also nur $q_1$ stehen. Dies ist zunächst einmal möglich. Dann ist zu unterscheiden:

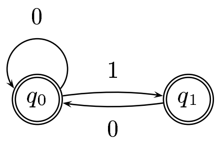



Fall 2 würde den Fall $q_1\, 1\, q_1\, L$ oder $q_1\, 0\, q_2\, R$ benötigen, der existiert. Um aber in diesen Zustand zu kommen, würde eine Regel $q_?\, 0 q_1 R$ benötigt werden, die nicht vorhanden ist.

also keine Länge von 3, also nur $[q_0], [q_1]$.

| | 0 | 1 |
|---|---|---|
| $[q_0]$ | $[q_0]$ | $[q_1]$ |
| $[q_1]$ | $[q_0]$ | |

Die Folge 101101 ergibt also: $q_0 \overset{1}{\vdash} q_1 \overset{0}{\vdash} q_0 \overset{1}{\vdash} q_1 \overset{1}{\vdash}$ und den Automaten:




---

## 1.2 Automaten mit Ausgabe

Die Idee bei diesen Automaten besteht neben dem akzeptieren eines $w \in \Sigma^*$, aus einer Ausgabe $x \in \Delta^*$ mit $\Delta \cap \Sigma = \emptyset$. Es wird zwischen folgenden zwei Automaten unterschieden:

1. Dem **Moore Automaten**, welcher beim Erreichen eines Zustandes $q \in Q$ eine Ausgabe liefert.
2. Dem **Mealy Automaten**, welcher während eines Übergangs $q_1 \vdash q_2$ eine Ausgabe liefert.

**Definition 1.10 (Moore Automat)**

*1.* $M = (Q, \Sigma, \boldsymbol{\Delta}, \delta, \lambda, q_0)$ *mit* $\delta : Q \to \Delta$

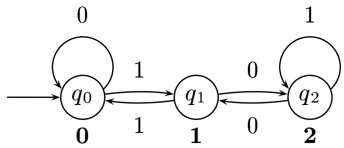
*2.* $w = a_1 a_2 \ldots a_n \in \Sigma^*$ *und* $q_o \overset{a_1}{\vdash} q_1 \overset{a_2}{\vdash} q_2 \vdash \ldots \overset{a_n}{\vdash} q_n$
*Mit der Ausgabe* $x = \lambda(w) := \lambda(q_0)\lambda(q_1)\ldots\lambda(q_n)$.
*Weiterhin gilt* $\lambda(\epsilon) := q_0$

**Beispiel**

Eingabe von $i \in N$ mit $i = a_1 a_2 \ldots a_n$, $a_i \in \{0, 1\}$ und Ausgabe von $i \bmod 3$.
Dann entspricht $a_1 \ldots a_n 0 = 2 \cdot i$ einer Verdoppelung und $a_1 \ldots a_n 1 = 2 \cdot i + 1$.
Betrachte folgende Tabelle:

| $i \bmod 3$ | $I$ | $i0$ | $i0 \bmod 3$ | $i1$ | $i1 \bmod 3$ |
|---|---|---|---|---|---|
| 0 | $3m$ | $6m$ | 0 | $6m+1$ | 1 |
| 1 | $3m+1$ | $6m+2$ | 2 | $6m+3$ | 0 |
| 2 | $3m+2$ | $6m+4$ | 1 | $6m+5$ | 2 |

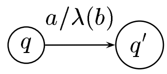
Es ergibt sich also folgender Automat (Beachte auch Quersumme 3):




---

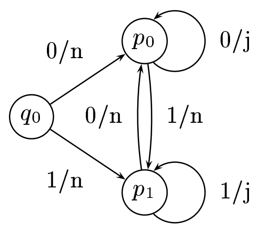
**Definition 1.11 (Mealy Automat)**

1. *$M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ mit $\lambda : Q \times \Sigma \to \Delta$*
2. *Sei $w = a_1 a_2 \dots a_n \in \Sigma^*$ und $\lambda(\epsilon) := \epsilon$*
   *Dann ist die Ausgabe $\lambda(w) := \lambda(q_0, q_1)\lambda(q_1, q_2) \dots \lambda(q_{n-1} q_n)$*




**Beispiel**

Sei $\Sigma = \{0, 1\}$, $\Delta = \{j, n\}$ und $c_0 = \{x00 | x \in \Sigma^*\} \cup \{x11 | x \in \Sigma^*\}$.
Der zugehörige Automat:




---

**Definition 1.12 (Äquivalente Automaten)**
*Sei $M_1$ ein Mealy Automat, $M_2$ ein Moore Automat, mit $w \in \Sigma^*$. und $\lambda_1(w)$, $\lambda_2(w)$ die Ausgabe durch $M_1$ bzw. $M_2$. Sei weiterhin $b_0$ die Ausgabe des $M_2$ bei $q_0$. Dann sind $M_1$ und $M_2$ äquivalent, wenn $b_0 \lambda_1(w) = \lambda_2(w)$ gilt.*

**Lemma 1.13** *Für jeden Moore Automaten $M$ existiert ein äquivalenter Mealy Automat $M'$.*

**Beweis**
$M'$ wie $M$ nur $\lambda'(q, a) = \lambda\big(\delta(q, a)\big)$. ■

**Lemma 1.14** *Für jeden Mealy Automaten $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ existiert ein äquivalenter Moore Automat $M' = (Q', \Sigma, \Delta, \delta', \lambda', q_0')$*

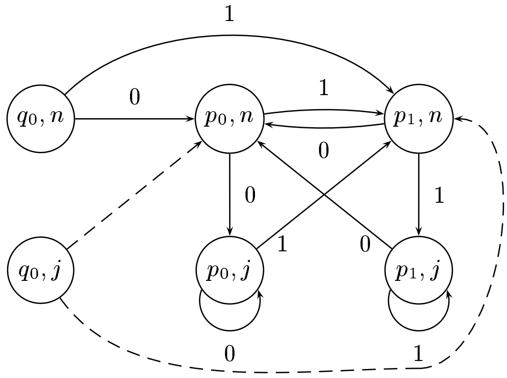
**Beweis**
Die Idee des Beweises besteht darin, das beim Mealy Übergang ausgegebene Zeichen zum Folgezustand hinzuzunehmen. Man erhält also einen $M'$-Zustand $[q, b]$ mit $q \in Q$, $b \in \Delta$, $Q' := Q \times \Delta$, und die Abbildung $\delta'\big([q, b], a\big) = \big[\delta(q, a), \lambda(q, a)\big]$ (also unabhängig von $b$) mit $\lambda'\big([q, b]\big) := b$ und $q_0' := [q_0, b_0]$, $b_0 \in \Delta$ beliebig.
Der endgültige Beweis wird dann mittels Induktion über die Länge der Ausgabe geführt. ■

**Beispiel**

Betrachte den Mealy Automaten aus dem letzten Beispiel. Man erhält also die Zustände

$$[q_0, j] \quad [q_0, n] \qquad [p_0, j] \quad [p_0, n] \qquad [p_1, j] \quad [p_1, j]$$

für $b_0 = n$ ergibt sich also folgender Automat:



1
0
1
0
0
1
1
0
0
1
$q_0, n$
$p_0, n$
$p_1, n$
$q_0, j$
$p_0, j$
$p_1, j$

# Kapitel 2

# Schaltwerke

## 2.1 Schaltnetze und Schaltwerke

Boole'sche Funktion $y = f(x_1, \dots, x_n)$ mit $x_i, y \in \{0,1\}$ und den Operationen $\wedge$ bzw. $\cdot$, $\vee$, $\neg$, $\not\equiv$ bzw. $+$ (antivalenz).

Desweiteren gilt das boole'sche Normalformentheorem:

$$\begin{aligned} f(x_1, \dots, x_n) &= (x_1 \wedge x_2 \wedge \dots \wedge x_n) \wedge f(1,1,\dots,1) \\ &= (\neg x_1 \wedge x_2 \wedge \dots \wedge x_n) \wedge f(0,1,\dots,1) \\ &= \dots \\ &= (\neg x_1 \wedge \neg x_2 \wedge \dots \wedge \neg x_n) \wedge f(0,0,\dots,0) \end{aligned}$$

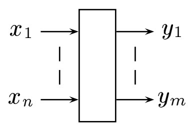
**Beispiel**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $x_1$ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| $x_2$ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| $x_3$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| $f$ | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |

also $f(x_1, x_2, x_3) = (x_1 \wedge \bar{x}_2 \wedge \bar{x}_3) \vee (\bar{x}_1 \wedge x_2 \wedge \bar{x}_3) \vee (x_1 \wedge \bar{x}_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge x_3)$

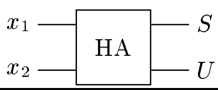
---

**Definition 2.1 (Schaltnetz)**
*Ein **Schaltnetz** besitzt $m$ Schaltfunktionen (boole'sche Funktionen) mit jeweils $n$ Variablen.*



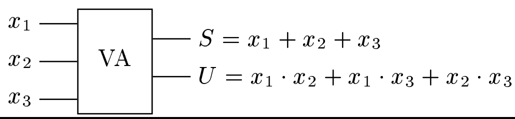

**Beispiel**

Halbaddierer und Volladdierer




$S = x_1 + x_2$ Summe

$U = x_1 \cdot x_2$ Übertrag

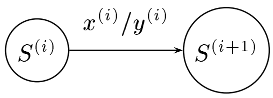



---

**Definition 2.2 (Schaltwerk)**

*Ein **Schaltwerk** ist ein Schaltnetz mit Speicher. Ein **Speicher** besteht aus Speichereinheiten zu je 1 Bit. Diese besitzen die Werte $s_0, s_1, \dots, s_{r-1}$.* $\boxed{s_i}$
*Die Ausgabe eines Schaltwerks ist Takt gesteuert und geschieht zu den Zeitpunkten $t_0, t_1, \dots, t_n$. Die Eingabe ist definiert als $x^{(i)} := (x_1, \dots, x_n)^{(i)}$. Die Ausgabe $y^{(i)} := (y_1, \dots, y_m)^{(i)}$ ist abhängig von $x^{(i)}$ und dem Zustand $s^{(i)} := (s_0, \dots, s_{r-1})^{(i)}$ zum Zeitpunkt $t_i$.*

Es ergibt sich also ein Mealy Automat der folgenden Form:




Speziell gilt:

1. Der Automat hat $2^r$ Zustände.
2. $y^{(i)}, s^{(i+1)}$ entstehen aus $s^{(i)}, x^{(i)}$ über boole'sche Verknüpfungen.

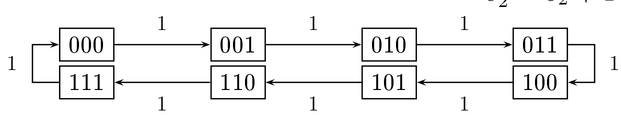
**Beispiel**

Dualzähler $D_3$ für 3 stellige Binärzahlen mit $s = s_0 s_1 s_2$. Mit Eingabe $e, s_i \in \{0,1\}$ und keiner Ausgabe.
Bei Eingeben von $e = 1$ sollen die Binärzahlen $000, 001, \dots, 111$ durchlaufen werden.
Es gilt $(s, e) \vdash s'$, $s = s_0 s_1 s_2$, $s' = s'_0 s'_1 s'_2$
Dann

$$s'_0 = s_0 + e s_1 s_2 \qquad s'_1 = s_1 + e s_2 \qquad s'_2 = s_2 + e$$

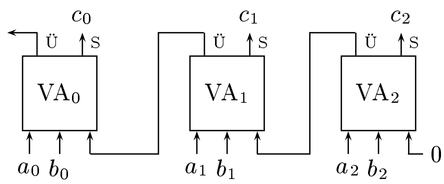
Also

$$e = 0 \Rightarrow s' = s \qquad\qquad e = 1 \Rightarrow \begin{aligned} s'_0 &= s_0 + s_1 s_2 \\ s'_1 &= s_1 + s_2 \\ s'_2 &= s_2 + 1 \end{aligned}$$



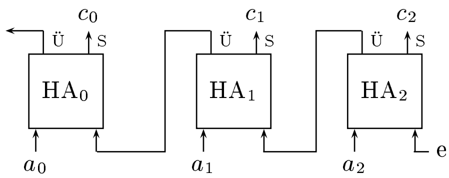

---

Veranschaulichung für die Übergangsregel des $D_3$. Start bei 000, mehrmaliges addieren von 001. Addiere $a_0 a_1 a_2 + b_0 b_1 b_2 = c_0 c_1 c_2$, also



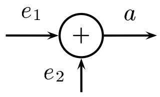

Für $b_0 b_1 b_2 = 00e$ ergibt sich eine Folge von Halbaddieren:

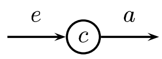
$S_j = a_j \quad S'_j = c_j$
$c_2 = a_2 + e \quad u_2 = a_2 \cdot e$
$c_1 = u_2 + a_1 = a_1 + a_2 e \quad u_1$
$u_1 = u_2 \cdot a_1 = a_2 a_1 e$
$c_0 = u_1 + a_0 = a_2 a_1 e + a_0$

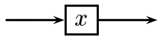



## 2.2 Schaltwerke für Polynome

Sei $a(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_{n-1} x^{n-1}$ mit $a_i \in \{0,1\}$. Die Bausteine für Schaltwerke sind dann:

1. Addierer $a = e_1 + e_2$ 

2. Multiplizierer mit einer Konstanten $c \in \{0, 1\}$, also $a = c \cdot e$ 

3. Speicherelemente mit $x \in \{0, 1\}$ 

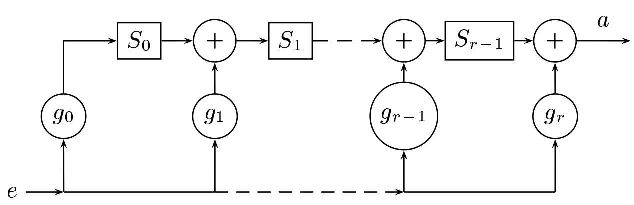

Bei 1. Takt geschieht also folgendes:

1. Der Inhalt jedes Speicherelements wird in Pfeilrichtung ausgegeben.
2. Alle Verknüpfungen werden ausgeführt.

Ein Polynom wird dann als $s(x) = s_0 + s_1 x + \cdots + s_{r-1} x^{r-1}$ dargestellt.

### 2.2.1 Der Multiplizierer

Betrachte folgendes Schaltwerk




Sei $s^{(0)}(x) = s_0^{(0)} + s_1^{(0)} x + \cdots + s_{r-1}^{(0)} x^{r-1}$ das zu untersuchende Polynom. Nacheinander eingeben von $e_0, e_1, \ldots, e_n$ (je 1 Wert pro Takt) liefert die Ausgabe $a_0, a_1, \ldots, a_n$.

**Lemma 2.3** *Bei Vorbesetzung $s_0^{(0)} = \cdots = s_{r-1}^{(0)} = 0$ entsteht durch die Eingabe von $e_0, e_1, e_n$ die Ausgabe $a_0, a_1, \ldots, a_n$ mit $a_j = e_j g_r + e_{j-1} g_{r-1} + \cdots + e_0 g_{r-j}$, wobei $j \in \mathbb{N}$ und $g_\nu = e_\nu = 0$ für $\nu < 0$.*

**Beweis**
Sei $s^{(j)}(x) = s_0^{(j)} + s_1^{(j)} x + \cdots + s_{r-1}^{(j)} x^{r-1}$, dann liefert die Eingabe $e_j$:

$$s_0^{(j+1)} = e_j g_0 \quad s_\nu^{(j+1)} = e_j g_\nu + s_{\nu-1}^{(j)} \quad \text{und} \quad a_j = e_j g_r + s_{\nu-1}^{(j)}$$

Weiterhin gilt:

$$\begin{aligned}
a_j &= e_j g_r + s_{\nu-1}^{(j)} \\
a_j &= e_j g_r + e_{j-1} g_{r-1} + s_{\nu-2}^{(j-1)} \\
a_j &= e_j g_r + e_{j-1} g_{r-1} + \cdots + e_1 g_{r-j+1} + s_{\nu-j}^{(1)} \\
a_j &= e_j g_r + e_{j-1} g_{r-1} + \cdots + e_1 g_{r-j+1} + e_0 g_{r-j} + \underbrace{s_{\nu-j-1}^{(0)}}_{=0}
\end{aligned}$$

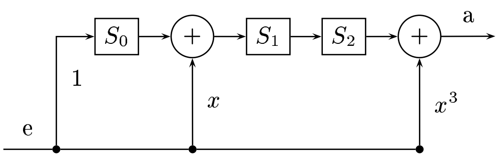
■

**Satz 2.4** *Sei $b(x) = b_0 + b_1 x + \cdots + b_k x^k$ und $g(x) = g_0 + g_1 s + \cdots + g_r x^r$.*
*Bei Eingeben von $b_k, \ldots, b_0, \underbrace{0, \ldots, 0}_{n-mal}$ und $s^{(0)}(x) = 0$ liefert das Schaltwerk die Koeffizienten $c_j$ von $c(x) = b(x)\dot{g}(x) = c_0 + c_1 x + \ldots c_{k+r} x^{k+r}$ als $c_{k+r}, c_{k+r-1}, \ldots, c_0$.*

**Beweis**
Obiges Lemma mit $k + r = n$, $e_j = b_{k-j}$ und $a_j = c_{k+r-j}$. ∎

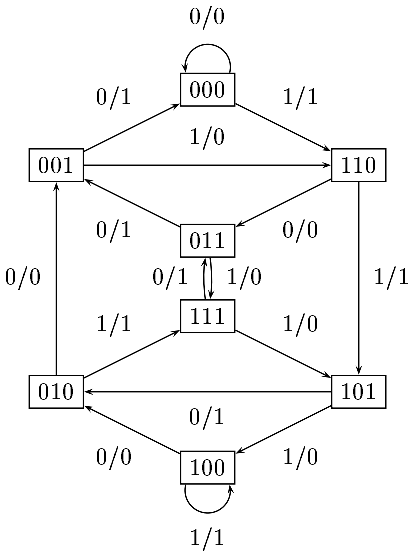
**Beispiel**

für $b(x) \cdot g(x)$ mit $g(x) = 1 + x + x^3$ betrachte folgendes Schaltwerk:




Es gilt also $S_0' = e$, $S_1' = e + S_0$, $S_2' = S_1$, $a0e + S_2$, wobei $e = b_k, b_{k-1}, \ldots, b_0, \underbrace{0, \ldots, 0}_{r\text{-mal}}$.

Es folgt also $c_{k+r}, c_{k+r-1}, \ldots, c_o$ für $c(X) = b(X) \cdot g(X)$
Hieraus kann folgender Automat abgeleitet werden:

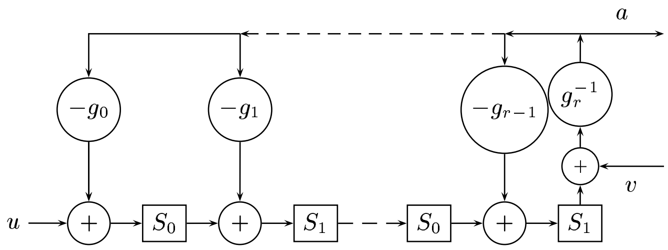



1. $b(x) = 1 + x^2 + x^3$, $g(x) = 1 + x + x^3$
   Eingabe: 1101:000 ($r = 3$ in $g(x)$, also 3 Nullen anhängen.
   Ausgabe: 1111111
   Also $c(x) = x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$
   Beachte auch den Zusammenhang zu den irredzuieblen Faktoren von $x^7 + 1 = (x + 1)(x^3 + x^2 + 1)(x^3 + x + 1)$ (Codierungstheorie).

2. Sei $b(x) = x + 1$, dann liefert die Eingabe 11000 die Ausgabe 11101.
   Man erhält also $c(x) = x^4 + x^3 + x^3 + 1$.

3. $b(x) = x^3$, also Eingabe 1000000 und Ausgabe 1011000

### 2.2.2 Der Modulierer




**Lemma 2.5** *Bei einem Takt entsteht in Schaltwerk B aus $s(x)$ ein $s'(x) = (x \cdot s(x) + v \cdot x^r + u) \bmod g(x)$*

**Beweis**
[verkürzt] Sei $s(x) = s_0 + s_1 x + \cdots + s_{r-1} x^{r-1}$ und $s'(x) = s'_0 + s'_1 x + \cdots + s'_{r-1} x^{r-1}$.
Also $s'_0 = u - g_0\big((v + s_{r-1}) g_r^{-1}\big)$ und $s'_i = s_{i-1} - g_1\big((v + s_{r-1}) g_r^{-1}\big)$.
Somit $s'(x) = \sum_{i=0}^{r-1} s'_i x^i = \cdots = \underbrace{x \cdot s(x) + vx^r + u}_{h(x)} - \underbrace{(v + s_{r-1}) g_r^{-1}}_{\text{Konstante } c} \cdot g(x)$

Also $h(x) = c \cdot g(x) + s'(x)$ mit $Grad\big(s'(x)\big) \leq r - 1$ ∎

**Satz 2.6** *Sei $d(x) = d_0 + d_1 x + \cdots + d_{n-1} x^{n-1}$ und $g(x) = g_0 + g_1 x + \cdots + g_r x^r$ mit $r \geq 2$. Bei einer Vorbesetzung $s^{(0)}(x) = 0$ und einer Eingabe von $u = d_{n-1}, \ldots, d_0$ (sowie stets $v = 0$) enthält das Register $s(x) = s^{(n)}(x) = d(x) \bmod g(x)$. ($s^{(n)}$ bezeichnet dabei den $n$-ten Schritt.)*

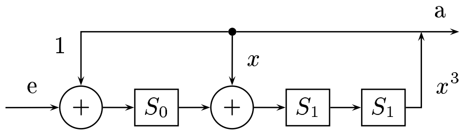
**Beweis**

1. $d^0(x) := d_{n-1}$, $d^{(i)}(x) = d_{n-i-1} + x \cdot d^{(i-1)}(x)$ für $i = 1, 2, \ldots, n-1$. Weiterhin gilt $d^{(n-1)}(x) = d(x)$
2. $s^0(x) = 0$. Bei Eingabe von $u = d_{n-1}$ und $v = 0$ entsteht
   $s^{(1)} = \big(\underbrace{xs^{(0)}(x)}_{=0} + d_{n-1}\big) \bmod g(x) = d_{n-1} = d^{(0)}(x) \bmod g(x)$
   Induktionsannahme:
   Aus $s^{(i)}(x) = d^{(i-1)}(x) \bmod g(x)$, folgt
   $$\begin{aligned} s^{(i+1)}(x) &= \big(x \cdot s^{(i)}(x) + d_{n-i-1}\big) \bmod g(x) \\ &= \big(x \cdot \big(d^{(i-1)}(x) \bmod g(x)\big) + d_{n-i-1}\big) \bmod g(x) \\ &= \ldots \\ &= d^{(i)}(x) \bmod g(x) \end{aligned}$$
3. $s^{(n)}(x) = \underbrace{d^{(n-1)}(x)}_{d(x)} \bmod g(x)$

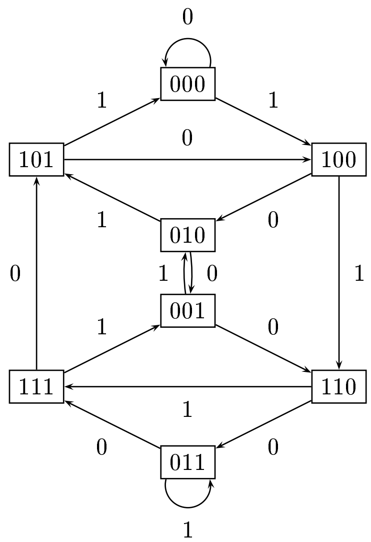
∎

**Beispiel**

Divisionsreste ( mod ) von $g(x) = x^3 + x + 1$. Hierzu wird folgendes Schaltwerk verwendet:




Für die Register ergibt sich also $S'_{=} = e + S_2$, $S'_1 = S_0 + S_2$, $S'_2 = S_1$.
Auch zu diesem Schaltwerk kann ein passender Automat konstruiert werden:




1. $d(x) = x^7 + 1$ Eingabe 10000001 liefert Ergebnis 000, also $g(x) \big| x^7 + 1$
2. $d(x) = x^6 + 1$ Eingabe 1000001 liefert Ergebnis 001, also $x^6 + 1 \bmod g(x) 0 x^2$
3. $d(x) = x^6 + x^2 + 1$ Eingabe 1000101 liefert Ergebnis 000, also $g(x) \big| \underbrace{x^6 + x^2 + 1}_{=(g(x))^2}$
4. $d(x) = x^4 + x^3 + x^2 + x$ Eingabe 11101 liefert Ergebnis 000, also $g(x) \big| d(x)$
5. $d(x) = g(x)$ Eingabe 1011 liefert Ergebnis 000, also $g(x) \big| g(x)$

# Kontextfreie (Typ 2) Sprachen

Grammatik $G = (V, T, P, S)$. $P : A \to \beta$ Produktionen mit $A \in V$ und $\beta \in (V \cup T)^*$.

1. einfache Umformungen:
   $A$ ist **erreichbar**, wenn $S \overset{*}{\Rightarrow} \alpha A \gamma$
   $A$ ist **nützlich**, wenn $A \overset{*}{\Rightarrow} x \in T^*$
2. Beseitigen von unerreichbaren oder unnützem $A$, wie $A \to \epsilon$ und Kettenproduktionen $A \to B$.

Unter einer **Linksableitung** (leftmost = lm) versteht man: $S \underset{lm}{\overset{*}{\Rightarrow}} x \in T^*$.

Ein $x \in L(G)$ (Sprachschatz) heißt genau dann **mehrdeutig**, wenn für $x$ zwei verschiedene Linksableitungen existieren.

Eine Grammatik $G$ heißt genau dann **mehrdeutig**, wenn in $L(G)$ mindestens ein mehrdeutiges $x$ (gemäß $G$) existiert.

**Beispiel**

syntaktische Mehrdeutigkeit in Programmiersprachen:
`if` Bedingung `then` `if` Bedingung `then` Anweisung `else` Anweisung
kann interpretiert werden als:
`if` Bedingung `then` ( `if` Bedingung `then` Anweisung `else` Anweisung )
`if` Bedingung `then` ( `if` Bedingung `then` Anweisung ) `else` Anweisung

---

Eine Sprache $L_0 \subseteq T^*$ heißt **inhärent mehrdeutig**, wenn für jedes $G$ mit $L_0$ der Sprachschatz $L(G)$ mehrdeutig ist.

# Kapitel 3

# Inhärente Mehrdeutigkeit bei kontextfreien Sprachen

**Satz 3.1** $L_0 = \left\{a^i b^i c^j \middle| i,j \geq 1\right\} \cup \left\{a^i b^j c^j \middle| i,j \geq 1\right\}$ *ist inhärent mehrdeutig.*

**Beweis**
Seien in $G$ weder unerreichbare noch nutzlose Symbole.
Eine Grammatik $G$ ist eine **Schleifengrammatik** , falls:

1. jedes $A \in V, A \neq S$ ist rekursiv, also $A \overset{*}{\Rightarrow} xAy$ mit $|xy| \geq 1$.
2. $S$ ist entweder rekursiv $S \overset{*}{\Rightarrow} xSy$ mit $|xy| \geq 1$, oder nur der Anfang einer Ableitung

Dann gilt: zu jeder eindeutigen Typ 2 Grammatik existiert eine eindeutige äquivalente Schleifengrammatik (o. Beweis). ∎

**Hilfssatz**

1. Für jedes $A \neq S$ gilt exklusiv entweder

$$\begin{array}{lll} A \overset{*}{\Rightarrow} xAy,\ (xy = a^m \vee xy = c^m) & \text{(Art-1)} & \text{oder} \\ A \overset{*}{\Rightarrow} a^m A b^m & \text{(Art-2)} & \text{oder} \\ A \overset{*}{\Rightarrow} b^m A c^m & \text{(Art-3)} & \end{array}$$

2. $S$ kommt nur einmal in einer Ableitung eines $x \in L$ vor.
3. Kein $x \in L_0$ mit nur Art-1 Ableitungen enthält mehr als $l$ $b$'s (Notation: $|x|_b \leq l$).

Hieraus folgt die Mehrdeutigkeit von $L_0$, denn es gilt:

**Behauptung**
*Für $p > l$ ist $a^{2p}b^{2p}c^{2p} \in L_0$ mehrdeutig.*

**Beweis**

1. Sei $z_1 := a^p b^p c^{2p} \in L_0$

   **Behauptung**
   *Dann tritt bei keiner Ableitung des $z_1$ ein nonterminal Art-3 auf.*
   Sei $S \overset{*}{\Rightarrow} w_1 A w_2 \overset{*}{\Rightarrow} w_1 w w_2 = a^p b^p c^{2p}$ mit $A \overset{*}{\Rightarrow} b^n A c^n$, dann

$S \stackrel{*}{\Rightarrow} w_1 A w_2 \stackrel{*}{\Rightarrow} w_1 b^p A c^p w_2 \stackrel{*}{\Rightarrow} w_1 b^p w c^p w_2 =: z'$, also $|z'|_a = p$, $|z'|_b = 2p$, $|z'|_c = 3p$ und somit $z' \in L_0$ im Widerspruch zur Behauptung.

In jeder Ableitung des $z_1$ existiert also höchstens ein Art-1 und Art-2.

**Behauptung**
*Für $p > l$ existiert nicht nur ein Art-1.*
D.h. in jeder Ableitung des $z_1$ existiert mindestens einmal ein Art-2:
Sei $S \stackrel{*}{\Rightarrow} w_1 A w_2 \stackrel{*}{\Rightarrow} w_1 w w_2 = z_1$ mit $A \stackrel{*}{\Rightarrow} a^n A b^n$, dann
$S \stackrel{*}{\Rightarrow} w_1 A w_2 \stackrel{*}{\Rightarrow} w_1 a^p A a b^p w_2 \stackrel{*}{\Rightarrow} w_1 a^p w b^p w_2 =: z_0$, mit $|z_0|_a = 2p$, $|z_0|_b = 2p$, $|z_0|_c = 2p$ und wegen $S \stackrel{*}{\Rightarrow} z_0$ gilt $z_0 = a^{2p} b^{2p} c^{2p}$. Für $a^{2p} b^{2p} c^{2p}$ existiert also eine Ableitung ohne Art-3, aber mit mindestens einem Art-2.

2. Zeige analog, das für $z_2 := a^{2p} b^p c^p \in L_0$ eine Ableitung des $z_0$ existiert, die zwar kein Art-2, aber mindestens ein Art-3 besitzt.

Es existieren also 2 verschiedene (Links-) Ableitungen für $z_0$ ∎

**Zum Hilfssatz** Punkt 1
$S \stackrel{*}{\Rightarrow} w_1 A w_2 \stackrel{*}{\Rightarrow} w_1 w w_2 = a^r b^s c^t$ mit $r = s \vee s = t$, dann gilt auch $S \stackrel{*}{\Rightarrow} w_1 x^q A y^q w_2 \stackrel{*}{\Rightarrow} w_1 x^q w y^q w_2$. Für $x$ und $y$ existieren dann folgende Möglichkeiten:

1. $x = a^m \vee x = b^m \vee x = c^m$ (analog für $y$)
2. $x = c^m \Longrightarrow y = c^n$
3. **Behauptung**
   $x = b^m \Longrightarrow y = c^n$
   **Beweis**
   $x = b^m \Longrightarrow y = b^n \vee y = c^n$ ∎
   **Behauptung**
   *In $xy$ existieren nicht nur $b$'s*
   **Beweis**
   $x = b^i, y = b^j \Longrightarrow S \stackrel{*}{\Rightarrow} w_1 b^{iq} w b^{jq} w_2 =: z_1$,
   für große $q$ ist $|z_1|_b > |z_1|_a$, $|z_1|_b > |z_1|_c$ also $z_1 \neq L_0$ ∎

4. **Behauptung**
   $x = a^m \Longrightarrow y = a^n \vee y = b^m$
   **Beweis**

   (a) **Behauptung**
   $y \neq c^n$
   **Beweis**
   Sei $y = c^n$ mit $n \geq 1$, dann gilt:
   $S \stackrel{*}{\Rightarrow} w_1 a^{mq} w c^{nq} w_2 =: z_2$ (wobei $w_1 w w_2 = z_0 = a^r b^s c^t$ mit $r = s \vee s = t$)
   Also $|z_2|_a = r + mq$, $|z_2|_b = s$ und $|z_2|_c = t + nq$. Für große $q$ ist aber $s \neq r + mq$ und $s \neq t + nq$, folglich ist $z_2 \neq L_0$ im Widerspruch zur Annahme und es gilt die Behauptung. ∎

(b) **Behauptung**
*$y \neq b^n$ für $n \neq m$*
**Beweis**
Sei $y = b^n$, dann $S \overset{*}{\Rightarrow} w_1 a^{mq} w b^{nq} w_2 =: z_3$, also $|z_3|_a = r + mq$, $|z_3|_b = s + nq$ und $|z_3|_c = t$.
Für $r = s$ gilt also $|z_3|_a \neq |z_3|_b$ und $|z_3|_b \neq |z_3|_c$ im Widerspruch zur Annahme.
Für $s = t$ und $q \geq 1$ gilt $|z_3|_b \neq |z_3|_c$ und $|z_3|_a \neq |z_3|_b$.
Nur bei $r + mq = s + nq \iff q = \frac{r-s}{m-n}$ (mit $r \neq s$) führt dies zu einem Widerspruch. ∎

∎

# Kapitel 4

# Syntaktische Analyse kontextfreier Sprachen

## 4.1 Lexikalische Analyse

**Definition 4.1** *Ein **Token** ist ein terminales Zeichen einer kontextfreien Grammatik (CFG (context free grammar)) (einer Programmiersprache).*

**Definition 4.2** *Ein **Lexem** ist eine Folge von Zeichen.*

$\boxed{\textbf{Beispiel}}$

In Pascal sind z.B. `var, for, while, x, y1, 3, 3.13` Lexeme

---

Die syntaktische Analyse benötigt einen String aus Tokens. Die Zuordnung der Lexeme zu den Tokens ist die Aufgabe der lexikalischen Analyse. Gesucht werden dabei möglichst lange Lexeme, welche durch “ ”,“,”,“.”,“;”,. . . beendet werden. Für diese Aufgabe werden meist endliche Automaten verwendet.

$\boxed{\textbf{Beispiel}}$

Gegenbeispiel in Fortran wäre $\underbrace{\text{DO 4 I}}_{\text{real-Variable}} = 1.25$ und $\underbrace{\text{DO 4 I} = 1{,}25}_{\text{Laufanweisung}}$ zu was?

---

$\boxed{\textbf{Beispiel}}$

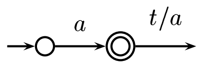
Eine Zahl in PASCAL (ohne Vorzeichen) besteht aus $z^+[.z^+]\big[E[+|-]z^+\big]$ mit $z$ Ziffer und $t$ Trennzeichen

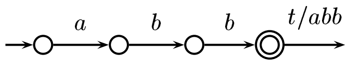
---

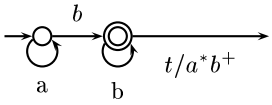
Eventuell bietet sich aber eine einfachere Vorgehensweise bei mehreren Varianten an. Konstruiere zuerst für jede Variante einen DFA, fasse diesen zu einem $\epsilon$DFA zusammen und wandle diesen in einen DFA um.

$\boxed{\textbf{Beispiel}}$

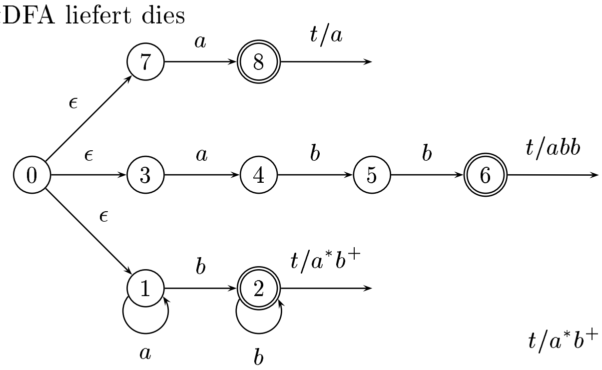
3 reguläre Ausdrücke mit $\Sigma = \{a, b\}$:

1. $a$ 

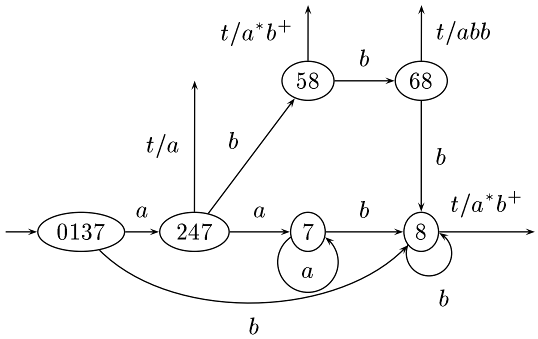
2. $abb$ 


3. $a^*b^+$ 


Zusammengefasst zu einem $\epsilon$DFA liefert dies




Umgewandelt in einen DFA:




$$
\begin{aligned}
\delta(0137, a) &= \delta(0, a) \cup \delta(1, a) \cup \delta(1, a) \cup \delta(3, a) \cup \delta(7, a) \\
&= \{2, 4, 7\} \cup \{2\} \cup \{4\} \cup \{7\} \\
&= \{2, 4, 7\}
\end{aligned}
$$

---

## 4.2 Syntaktische Analyse (Grundlagen)

Sei eine kontextfreie Grammatik $G$ gegeben und $x \in T^*$. Dann stellen sich folgende Aufgaben:

1. Gilt $x \in L(G)$? (Wortproblem)
2. Falls $x \in L(G)$ gilt, welche Linksableitung(en) gibt es für $x$? (Analyseproblem)[1]

**Definition 4.3** *Ein **Kellerautomat** (PDA push down automaton) besitzt einen Übergang:* $\left(q, \left\{ \begin{smallmatrix} a \\ \epsilon \end{smallmatrix} \right\}, z\right) \vdash \{(q', \gamma), \dots\}$ ***(NPDA)**, bzw.* $\left|\delta(q, a, z)\right| + \left|\delta(q, \epsilon, z)\right| \leq 1$ ***(DPDA)***
*Die Konfiguration eines Kellerautomaten besteht aus $(q, y, \beta)$, wobei $y$ der Rest auf dem Leseband und $\beta$ den Kellerinhalt darstellt.*

Ein Wort $w$ kann auf zwei Arten akzeptiert werden.

1. Über dem leeren Keller, also
   $(q_0, w, z_0) \vdash^* (q, \epsilon, \epsilon)$ mit $q$ unerheblich. Dann ist $N(M)$ die akzeptierte Sprache.
2. Über Endzustand
   $(q_0, w, z_0) \vdash (q_f, \epsilon, \beta)$ mit $q_f \in F$ und $\beta$ unerheblich. Dann ist $T(M)$ die akzeptierte Sprache.

Zu jeder kontextfreien Grammatik $G$ existiert ein NPDA $M\big[L(G) = N(M)\big]$, mit $G = (V, T, P, S)$, $P = \{A \to \beta\}$ und $M = \big(\{q_0\}, T, V \cup T, \delta, q_0, S, \phi\big)$. (Wobei $T$ dem Bandalphabet $\Sigma$, $V \cup T$ dem Kelleralphabet $\Gamma$ und $S$ dem ? $z_0$ entspricht).

[1] Ein **Parser** ist ein Verfahren zur Lösung dieses Problems

Es gilt:

$$\delta(q_0, a, Z) = \begin{cases} (q_0, e) & \text{für } Z = a \\ \phi & \text{sonst} \end{cases} \qquad \delta(q_0, \epsilon, Z) = \begin{cases} \phi & \text{für } Z \notin V \\ \{(q_0, \beta) \big| Zto\beta \in P\} & \text{für } Z \in V \end{cases}$$

**Beispiel**

$S \to bDAe$, $D \to dD \mid d$, $A \to aA \mid a$, also $(bd^+a^+e)$.

1. $(q_0, a, a)$
2. $(q_0, b, b)$
3. $(q_0, d, d)$
4. $(q_0, e, e)$
5. $(q_0, \epsilon, S)$ $\quad (q_0, bDAe)$
6. $(q_0, \epsilon, D)$ $\quad (q_0, dD), (q_0, d)$
7. $(q_0, \epsilon, A)$ $\quad (q_0, aA), (q_0, a)$

oder

| $bddae$ | $S$ |
|---|---|
| $\cancel{b}ddae$ | $\cancel{b}DAe$ |
| $ddae$ | $DAe \vdash ddae, dAe \vdash \ldots \natural$ |
| $\cancel{d}dae$ | $\cancel{d}DAe$ |
| $dae$ y | $DAe \vdash \ldots \natural$ |
| $\cancel{d}ae$ | $\cancel{d}Ae$ |
| $ae$ | $Ae \vdash \ldots \natural$ |
| $\cancel{a}e$ | $\cancel{a}e$ |
| $\cancel{e}$ | $\cancel{e}$ |
| $\epsilon$ | $\epsilon$ (akzeptiert) |

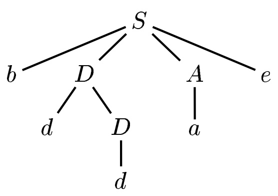
Wobei die Fälle 1 bis 4 $(q_0, \epsilon)$ entsprechen.

### 4.2.1 Top-Down-Analyse

$(x, S) \overset{*}{\Rightarrow} (\epsilon, \epsilon)$ über

1. $(ay, a\gamma) \vdash (y, \gamma$ mit $a \in T$
2. $(y, A\gamma) \vdash (y, \beta\gamma)$ mit $(A \to \beta) \in P$

Zum Beispiel




- auch möglich (von $ddae, DAe$)
  $ddae\,,\ DAe \vdash ddae\,,\ dAe \vdash dae\,,\ Ae$
  $\vdash$ entweder $dae\,,\ a\epsilon$ oder $dae\,,\ aA$
- auch möglich (von $dae, DAe$)
  $dae\,,\ DAe \vdash dae\,,\ dDAe \vdash ae\,,\ DAe$
  $\vdash$ entweder $ae\,,\ dae\ \natural$ oder $ae\,,\ dDae\ \natural$
- auch möglich (von $ae, Ae$)
  $ae\,,\ Ae \vdash ae\,,\ aAe \vdash e\,,\ Ae$
  $\vdash$ entweder $e\,,\ ae\ \natural$ oder $e\,,\ aAe\ \natural$

### 4.2.2 Zweite Technik: Bottom-up-Analyse

Die Idee dieser Technik, ist eine Suche in $x \in T^*$ nach Teilstrings $\beta$ mit $A \to \beta \in P$. Diese $\beta$ in $x$ werden solange durch $A$ ersetzt, bis $S$ entsteht.

**Beispiel**

Sei $G_{arith}$ (arithmetisch) eine Grammatik mit Startsymbol $E$, dann: $E \to E + T \mid T \qquad T \to T\,*$

$F \mid F \qquad F \to (E) \mid a$

$$
\begin{array}{ll}
x = a + a * a + a \quad \vdash & \\
\quad F + a * a + a \quad \vdash & \\
\quad T + a * a + a \quad \vdash & \\
\quad E + a * a + a \quad \vdash & \\
\quad E + T * a + a & \vdash E * a + a \quad \vdash \quad \natural \\
\quad E + T * F + a & \vdash E * F + a \quad \vdash \quad \natural \\
\quad E + T * a \quad \vdash & \\
\quad E + a \quad \vdash & \\
\quad E + F \quad \vdash & \\
\quad E + T \quad \vdash & \\
\quad E &
\end{array}
$$

---

Suche also nach Unterklassen der Typ-2-Sprachen, bei denen möglichst wenig (oder keine) Sackgassen bei der syntaktischen Analyse auftreten. Diese Klassen heißen **LL** und **LR**.

## 4.3 Sackgassenfreie Analyse durch Links- und Rechts-kontext

**Definition 4.4** *Eine Grammatik $G$ heißt genau dann* ***sackgassenfrei****, wenn für $x \in L(G)$ und $\alpha \overset{*}{\Rightarrow} x$ bezüglich bottum-up $S \overset{*}{\Rightarrow} \alpha$ gilt.*

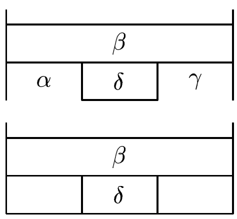
**Definition 4.5** *$\beta \in \left(V \cup T\right)^*$ heißt genau dann* ***lokale Mehrdeutigkeit****, wenn für $\beta = \alpha\delta\gamma$ entweder*

*1. $A \to \delta$, $B \to \beta$ mit $A \neq B$ oder $\delta \neq \beta$*

*oder 2. $A \to \alpha\gamma$, $B \to \delta\gamma$ mit $\gamma \neq \epsilon$*

*gilt.*

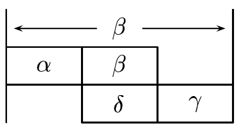
1.




$E \to \underbrace{E+T}_{\beta} \qquad E \to \underset{\delta}{\underset{\uparrow}{T}}$

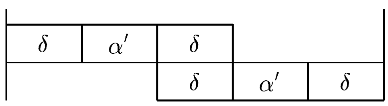
$A + B$

2.




$E \to \underbrace{E+}_{\alpha}\underbrace{T}_{\delta} \qquad T \to \underbrace{T}_{\delta}\underbrace{*F}_{\gamma}$

$\beta = \underbrace{E+}_{\alpha}\underset{\delta}{T} * \underbrace{T}_{\delta}$




**Lemma 4.6** *Eine CFG ohne lokale Mehrdeutigkeit ist sackgassenfrei (und eindeutig)!*

**Beweis**

1. Annahme: Es existiert keine lokalen Mehrdeutigkeit.
   Dann $S \overset{*}{\Rightarrow} \phi$ und $\beta \overset{*}{\Rightarrow} \varphi$, also $S \overset{*}{\Rightarrow} \beta$.
   Die Ableitung $\delta' : S \Rightarrow \varphi_1 \Rightarrow \cdots \Rightarrow \varphi_n = \varphi$;
   $\beta \Rightarrow \varphi$: dann $\beta = \gamma A \delta$, $\varphi = \gamma\delta\varrho\delta$, $A \to \varrho$
   Da keine lokalen Mehrdeutigkeit bleibt rückwärts in $\lambda$ zunächst erhalten und wird beidseitig nur durch $\varrho \vdash A$ ersetzt.
   Sei $\varrho \subseteq \varphi_i$ für $i = r, \ldots, n$ mit $\varrho \nsubseteq \varphi_{r-1}$.
   Dann $\varphi_r = \gamma'\varrho\delta'$ mit $\gamma' \overset{*}{\Rightarrow} \gamma$, $\delta' \overset{*}{\Rightarrow} \delta$, $\varphi_{r-1} = \gamma' A \delta'$.
   Damit dann $S \overset{*}{\Rightarrow} \varphi_{r-1} = \gamma' A \delta' \overset{*}{\Rightarrow} \gamma A \delta = \beta$ bzw. $S \overset{*}{\Rightarrow} \varphi_{r-1} = \gamma' A \delta' \overset{*}{\Rightarrow} \gamma'\varrho\delta' = \gamma\varrho\delta = \varphi$
2. $x \in L(G)$, $\alpha \overset{*}{\Rightarrow} x$, dann $\alpha \Rightarrow \alpha_1 \Rightarrow \cdots \Rightarrow \alpha_m \Rightarrow x$, $S \overset{*}{\Rightarrow} x$, $\alpha_m \Rightarrow x$.
   Damit (wegen 1.) $S \overset{*}{\Rightarrow} \alpha_m$ usw.
   Insgesamt also $S \overset{*}{\Rightarrow} \alpha$

■

### 4.3.1 Verminderung der Anzahl der Sackgassen

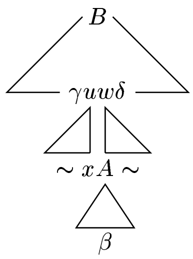
Die Idee stammt von Eichel und Paul (1964). Sei $A \to \beta$ eine Produktion. Die Reduktion $\alpha_1\beta\alpha_2 \vdash \alpha_1 A \alpha_2$ ist nur möglich für ein spezielles Ende von $\alpha_1$. bzw. Anfang des $\alpha_2$.
Z.B. $B \to \beta_1 CAD\beta_2$, dann ist das erste Zeichen des $\alpha_2$ der Anfang eines aus $D$ erzeugten Strings.
Das letzte Zeichen des Strings des $\alpha_1$ ist das Ende eines aus $C$ erzeugten Strings, d.h. $\beta \vdash A$ nur möglich in gewissen “Kontexten”.

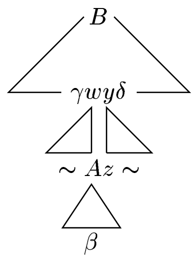
**Definition 4.7** *$x \in V \cup T$, dann*

$$\alpha(x) := \left\{Y \in V \cup T \middle| x \overset{*}{\Rightarrow} Y\beta\right\} \qquad w(x) := \left\{Y \in V \cup T \middle| x \overset{*}{\Rightarrow} \beta Y\right\}$$

*speziell gilt: $X \in \alpha(x)$, $x \in w(x)$ und $x \in T \Longrightarrow \alpha(x) := w(x) = \{x\}$*
$\gamma x \beta z \delta \vdash \gamma x A z \delta,\ A \to \beta$
*Ferner existieren noch die Randzeichen $\# \in T$ und die Reduktion auf $\#S\#$.*

$$L_A = \left\{X \middle| B \to \gamma u w \delta, x \in w(u), A \in \alpha(w)\right\} \qquad R_A = \left\{Z \middle| B \to \gamma w y \delta, z \in \alpha(\gamma), A \in w(w)\right\}$$




*Weiterhin gilt bei $A \in \alpha(S)$ ist $\# \in L_A$ und bei $A \in w(S)$ ist $\# \in R_A$.*

**Beispiel**

$G_{\text{arith}}$, $E \to E + T$, $E \to T$, $T \to T * F$, $T \to F$, $F \to (E)$, $F \to a$.

Lokale Mehrdeutigkeiten

$$T * F \qquad E + T \qquad E + T * F$$

---

**Beispiel** **Anmerkung**

$L_T$ für $T * F \vdash T$ und $F \vdash T$

1. Berechne $\alpha$- und $w$-Klassen

| | $\alpha$ | $w$ |
|---|---|---|
| $E$ | $ETFa($ | $ETFa)$ |
| $T$ | $TFa($ | $TFa)$ |
| $F$ | $Fa($ | $Fa)$ |

2. Bestimme alle $W$ mit $T \in \alpha(W) : E, T$. Betrachte alle Reduktionen mit $E$ bzw. $T$ rechts: Was steht links davon?

Links von $E$ stehen nur (, links von $T$ nur + (oder $t$?).
Zusätzlich $T \in \alpha(E) \Longrightarrow$ noch #, d.h. $L_T = \{(, +, \#\}$.
Folglich ist die Reduktion $\beta \vdash T$ nur möglich, falls links von $\beta$ ein Zeichen aus $L_T$ steht. Z.B.: $T * F \vdash T * T$

$$L_T : +(\# \qquad\qquad L_E : (\#$$
$$R_T : +*)\# \qquad\qquad R_E : +)\#$$

$\beta \vdash T$ für $\begin{array}{c|c|c} + & & + \\ & & * \\ ( & \beta & ) \\ \# & & \# \end{array}$ $\quad \beta \vdash E$ für $\begin{array}{c|c|c} & & + \\ ( & \beta & ) \\ \# & & \# \end{array}$ $\quad \#E + \underline{T * F}\# \quad \#\underline{E + T}\# \quad E$

---

## 4.4 Operator- und Präzedenzsprachen

**Definition 4.8** *Eine CFL heißt* ***Operatorsprache****, wenn hierfür eine Grammatik existiert, welche keine Produktion der Form $A \to \gamma BC\delta$ mit $B, C \in X$ besitzt.*

**Definition 4.9 (Phrase, Primphrase)** *$\gamma$ ist genau dann eine* ***Phrase****, wenn $S \overset{*}{\Rightarrow} \alpha A \beta$ und $A \overset{*}{\Rightarrow} \gamma$.*

*$\gamma$ ist eine Primphase, genau dann wenn $\gamma$ eine Phrase ist, in $\gamma$ terminale Symbole [sind?], und keine echte Teilphrase von $\gamma$ enthält terminale Symbole.*

**Beispiel**

$G_{\text{arith}}$: $S \overset{*}{\Rightarrow} \underbrace{F + F}_{\gamma_1} * (\underbrace{F + F}_{\gamma_1}) \overset{*}{\Rightarrow} a + a * (a + a)$ (wobei $F * (F + F) = \gamma_2$)

$\gamma_1$ Phrase: $E \overset{*}{\Rightarrow} \gamma_1$, Primphrase: Teilphrase nur $F$ (denn $E \overset{*}{\Rightarrow} F$).
$\gamma_2$ Phrase: $E \Rightarrow T \overset{*}{\Rightarrow} \gamma_2$, keine Primphrase.

---

**Definition 4.10** *Sei $a, b \in T$, dann*

1. $a \doteq b \quad \Longleftrightarrow \quad A \to \alpha ab\gamma$ *oder* $A \to \alpha aBb\beta$
2. $a \lessdot \quad \Longleftrightarrow \quad A \to \alpha aB\beta$ *und* $B \overset{*}{\Rightarrow} b\gamma$ *oder* $B \overset{*}{\Rightarrow} Cb\gamma$
   *D.h. beim erzeugen entsteht a vor b*

3. *$a \gtrdot b \iff A \to \alpha B b \beta$ und $B \overset{*}{\Rightarrow} \gamma a$ oder $B \overset{*}{\Rightarrow} \gamma a C$*
   *D.h. beim Erzeugen entsteht $b$ vor $a$*

**Definition 4.11 (Präzedenzsprache)** *Eine Operatorsprache heißt **Präzedenzsprache** genau dann wenn für $a, b \in T$ mit $S \overset{*}{\Rightarrow} \alpha a b \beta$ oder $S \overset{*}{\Rightarrow} \alpha a B b \beta$ es höchstens eine Relation $\doteq, \lessdot, \gtrdot$ existiert.*

***Präzedenzmatrix***

| | $a$ | ( | $*$ | $+$ | ) |
|---|---|---|---|---|---|
| $a$ | | | $\gtrdot$ | $\gtrdot$ | $\gtrdot$ |
| ( | $\lessdot$ | $\lessdot$ | $\lessdot$ | $\lessdot$ | $\doteq$ |
| $*$ | $\lessdot$ | $\lessdot$ | $\gtrdot$ | $\gtrdot$ | $\gtrdot$ |
| $+$ | $\lessdot$ | $\lessdot$ | $\lessdot$ | $\gtrdot$ | $\gtrdot$ |
| ) | | | $\gtrdot$ | $\gtrdot$ | $\gtrdot$ |

$$
\begin{aligned}
E &\to E + T \\
E &\to T \\
T &\to T * F \\
T &\to F \\
F &\to (E) \\
F &\to a
\end{aligned}
$$

**Lemma 4.12** *In einer Operatorsprache gilt für $a, b \in T$ mit $S \overset{*}{\Rightarrow} \pi a b \mu$ oder $S \overset{*}{\Rightarrow} \pi a D b \mu$ stets wenigstens eines aus $\doteq, \lessdot, \gtrdot$*

**Beweis**
$\delta_1 := \pi a b \mu$, $\delta_2 := \pi a D b \mu$, falls $A \to \alpha a b \beta$ oder $A \to \alpha a D b \mu$, ist $a \doteq b$.
Falls **nicht**, entsteht die Zeichen $b$ bzw. $Db$ rechts von $a$ und somit

**Fall 1**
$S \overset{*}{\Rightarrow} \pi a C \mu' =: \delta$, $C \overset{*}{\Rightarrow} b\delta$ oder $C \overset{*}{\Rightarrow} D b \delta$.
Es existiert Rand $A \to \alpha a B \beta$ mit $B \overset{*}{\Rightarrow} C \sim$, d.h. $B \overset{*}{\Rightarrow} b \sim$ oder $B \overset{*}{\Rightarrow} Db$, d.h. also $a \lessdot b$ ($a$ vor $b$).

**Fall 2** (analog)
$S \overset{*}{\Rightarrow} \lambda' C b \mu$, $C \overset{*}{\Rightarrow} \delta a$ bzw. $C \overset{*}{\Rightarrow} \delta a D$
also $a \gtrdot b$

■

**Lemma 4.13** *In einer Präzedenzsprache gilt für $a, b \in T$ mit $S \overset{*}{\Rightarrow} \pi a b \mu$ bzw. $S \overset{*}{\Rightarrow} \pi a D B \mu$ genau eine der folgenden Fälle:*

1. *Rechts von $a$ beginnt eine Phrase $b \sim$ bzw. $DB \sim$ ($a \lessdot b$).*
2. *Links von $b$ endet eine Phrase $\sim a$ bzw. $\sim aD$ ($a \gtrdot b$).*
3. *$a, b$ in der selben Phrase.*

Als direkte Folge aus obigem Lemma:

$$
\begin{aligned}
a \lessdot b : \pi a[b\mu \quad &\text{bzw.} \quad \pi a[Db\mu \\
a \gtrdot b : \pi a]b\mu \quad &\text{bzw.} \quad \pi aD]b\mu
\end{aligned}
$$

$\boxed{\textbf{Beispiel}}$

$G_{\text{arith}}$; $\#a + a * (a + a)\#$
$\#{}_{\lessdot}[a]_{\gtrdot} + {}_{\lessdot}[a]_{\gtrdot} * {}_{\lessdot}[({}_{\lessdot}[a]_{\gtrdot} + {}_{\lessdot}[a]_{\gtrdot})]_{\gtrdot}\#$
$\#\underset{\lessdot}{[}F + \underset{\lessdot}{[}F * \underset{\lessdot}{[}(\underset{\lessdot}{[}F + F\underset{\gtrdot}{]})\underset{\gtrdot}{]}\#$

$\#[F + [F * [(E)]\#$
$\quad\lessdot \qquad \lessdot \qquad \lessdot \doteq \gtrdot$
$\#[F + [F + F]\#$
$\quad\lessdot \qquad \lessdot \qquad \gtrdot$
$\#[F + T]\#$
$\quad\lessdot \qquad \gtrdot$
$\#E\#$

---

## 4.5 LL1 Sprachen

Top down Analyse, mit 1 Zeichen "Vorschau". Ab jetzt ein Randzeichen \$ rechts:

$$S\$ \overset{*}{\Rightarrow} wA\varphi \Rightarrow w\beta\varphi \overset{*}{\Rightarrow} waz =: z_1$$
$$S\$ \overset{*}{\Rightarrow} wA\varphi \Rightarrow w\gamma\varphi \overset{*}{\Rightarrow} wby =: z_2$$

Wobei $z$ und $y$ für einen Rest und $\varphi$ für beliebige Zeichen stehen.

**Definition 4.14** *Eine CFG G heißt* ***LL(1)*** *Grammatik, wenn aus*

$$S\$ \overset{*}{\Rightarrow} wA\varphi \Rightarrow w\beta\varphi \overset{*}{\Rightarrow} waz$$
$$S\$ \overset{*}{\Rightarrow} wA\varphi \Rightarrow w\gamma\varphi \overset{*}{\Rightarrow} way$$

*stets folgt, dass* $\beta = \gamma$.
*Eine Sprache heißt* ***LL1 Sprache*** *wenn eine LL1 Grammatik dafür existiert.*

**Definition 4.15** *Eine Menge* $E(A \to \beta)$ *heißt Menge der* ***Entscheidungszeichen*** *für* $A \to \beta$.

**Definition 4.16** *Eine Grammatik G ist genau dann vom Typ LL1, wenn bei* $A \to \beta$, $A \to \gamma$ *für* $\beta \neq \gamma$ *gilt:* $E(A \to \beta) \cap E(A \to \gamma) = \emptyset$

**Beispiel**

Sprache $bd^*a^*e$, $S \to bDAe$, $D \to dD \,|\, \epsilon$ und $A \to aA \,|\, \epsilon$. Dann:

| $A \to \beta$ | $F(A \to \beta)$ |
|---|---|
| $S \to bDAe$ | $b$ |
| $D \to \epsilon$ | $a, e$ |
| $D \to aD$ | $d$ |
| $A \to \epsilon$ | $e$ |
| $A \to aA$ | $a$ |

---

**Definition 4.17** *Sei G eine CGF, unter Umständen mit* $\epsilon$ *Regeln.*

1. $\text{first}(\beta) := \{a \,|\, \beta \overset{*}{\Rightarrow} a\varphi\}$
2. $\text{last}(B) := \{A \,|\, B \overset{*}{\Rightarrow} \varphi A\}$
3. $\text{follow}(A) := \{a \,|\, S\$ \overset{*}{\Rightarrow} wA\varphi A, a \in \text{first}(\varphi)\}$

**Lemma 4.18** *1.* first

(a) $\text{first}(\epsilon) := \emptyset$

(b) $\beta = a\varphi \implies \text{first}(\beta) = \{a\}$

(c) $\beta = X\varphi, X \in V \implies \text{first}(\beta) = \text{first}(X)$ *für* $X \overset{*}{\Rightarrow} \epsilon \cup first(\varphi)$ *für* $X \overset{*}{\Rightarrow} \epsilon$.
$\text{first}(x) : x\varrho Q$ *durch* $Xjkto\alpha Q\beta$, $\alpha \overset{*}{\Rightarrow} \epsilon \quad \text{first}(x) = \{a|X\varrho^+ a\}$

2. $\text{last}(B) = \{A \mid B \overset{*}{\Rightarrow} \varphi A\alpha, \alpha \overset{*}{\Rightarrow} \epsilon\}$
3. follow
   (a) $\$ \in \text{follow}(A) \iff A \in \text{last}(S)$
   (b) *bei* $a \neq \$$ *gilt:* $a \in \text{follow}(A) \iff D \to \alpha B\beta,\ A \in \text{last}(B),\ a \in \text{first}(\beta)$

$\boxed{\textbf{Beispiel}}$ **zu obigen Lemma**

Sei $V = \{S, A, A\}$, $S\varrho b$, $D\varrho d$, $A\varrho a$.
(last) $S \overset{+}{\Rightarrow} \varphi S$ kann nicht auftreten, also $A \overset{+}{\Rightarrow} a^n A$, $D \overset{+}{\Rightarrow} d^n D$ für $n \geq 1$.
$\text{follow}(S) : S \in \text{last}(S)$, $S$ steht nie rechts.
$\text{follow}(D) : D \in \text{last}(D)$, $D$ steht rechts bei:
1. $S \to bD\underbrace{Ae}_{\varphi}$ $\quad \text{first}(\varphi) = \underbrace{\text{first}(A)}_{a} \cup \{e\}$ $\qquad$ 2. $D \to dD\underset{\underset{\varphi}{\uparrow}}{\epsilon}$ $\quad \text{first}(\varphi) = \emptyset$

$\text{follow}(A) : A \in \text{last}(A)$, $A$ steht rechts bei:
1. $S \to bDA\underset{\underset{\varphi}{\uparrow}}{e}$ $\quad \text{first}(\varphi) = \{e\}$ $\qquad$ 2. $A \to aA\underset{\underset{\varphi}{\uparrow}}{\epsilon}$ $\quad \text{first}(\varphi) = \emptyset$

Es ergibt sich also folgende Tabelle:

| X | first($X$) | last($X$) | follow($X$) |
|---|---|---|---|
| $S$ | $b$ | $S$ | \$ |
| $D$ | $d$ | $D$ | $a, e$ |
| $A$ | $a$ | $A$ | $e$ |

---

**Lemma 4.19** *Die Entscheidungsmenge ist*

$$E(A \to \beta) = \begin{cases} \text{first}(\beta) & \textit{für } \beta \neq \epsilon \\ \text{follow}(A) & \textit{für } \beta = \epsilon \end{cases}$$

*bzw.* $E(A \to \beta) = \text{first}(\beta\,\text{follow}(A))$

$\boxed{\textbf{Beispiel}}$

| | | |
|---|---|---|
| $S \to bDAe$ | $\beta = bDAe$ | $\Longrightarrow \text{first}(\beta) = \{b\}$ |
| $D \to \epsilon$ | $\beta = \epsilon$ | $\Longrightarrow \text{follow}(D) = \{a, e\}$ |
| $D \to dD$ | $\beta = dD$ | $\Longrightarrow \text{first}(\beta) = \{a\}$ |
| $A \to \epsilon$ | $\beta = \epsilon$ | $\Longrightarrow \text{follow}(A) = \{e\}$ |
| $A \to aA$ | $\beta = aA$ | $\Longrightarrow \text{first}(\beta) = \{a\}$ |

---

$\boxed{\textbf{Beispiel}}$

Top-down Analyse mit Vorschau

| | |
|---|---|
| $\cancel{b}ddae$ | $S$ |
| | $\cancel{b}DAe$ |
| $\cancel{d}dae$ | $DAe$ |
| | $\cancel{d}DAe$ |
| $\cancel{d}ae$ | $DAe$ |
| | $\cancel{d}DAe$ |
| $\cancel{a}e$ | $DAe$ |
| | $Ae$ |
| | $\cancel{a}e$ |

---

Es gilt:

- Jede LL(k)-Sprache wird von einem DPDA über Endzustand akzeptiert.
- Die LL(1) Analyse ist "präfixkorrekt" (Sackgassenfrei). Aus $(XY, S) \overset{*}{\vdash} (Y, \varphi)$ folgt, dass ein $Z : XZ \in L(G)$ existiert.

- Der Zeitbedarf für eine Analyse ist $O(|X|)$.

**Algorithmus für LL(1) Analyse**

Mittels "rekursivem Abstieg" (recursive descent):
Für jedes Nonterminal eine (meist rekursive) Prozedur. Eine globale Variable $z$ für das nächste Eingabezeichen.

**Beispiel**

Prozedur `S` für obige Beispiele.

---

## 4.6 LR Sprachen

Lookahead rightmost derivation. Dieses Verfahren arbeitet bottom-up und von links nach rechts. Die Verarbeitung geschieht zeichenweise und eine Reduktion wird sobald wie möglich durchgeführt.

**Beispiel**

Grammatik $S' \to Sc$, $S \to SA \mid A$, $A \to aSb \mid ab$.
Dann $x = aababbc \vdash aAabbc \vdash aSabbc \vdash a\boxed{\text{SA}}bc \vdash aSbc \vdash Ac \vdash Sc \vdash S'$ (eingekastelt ist die lokale Mehrdeutigkeit). Mittels LR würde sich ergeben:

$$\begin{array}{llll}
(\epsilon\,,\,aababbc) & \overset{s}{\vdash} (a\,,\,ababbc) & \overset{s}{\vdash} (aa\,,\,babbc) & \overset{s}{\vdash} (aab\,,\,abbc) \\
\overset{r}{\vdash} (aA\,,\,abbc) & \overset{r}{\vdash} (aS\,,\,abbc) & \overset{s}{\vdash} (aSa\,,\,bbc) & \overset{s}{\vdash} (aSab\,,\,bc) \\
\overset{r}{\vdash} (aSA\,,\,bc) & \overset{r}{\vdash} (AS\,,\,bc) & \overset{s}{\vdash} (aSb\,,\,c) & \overset{r}{\vdash} (A\,,\,c) \\
\overset{r}{\vdash} (S\,,\,c) & \overset{s}{\vdash} (Sc\,,\,\epsilon) & \overset{r}{\vdash} (S'\,,\,\epsilon) &
\end{array}$$

---

Allgemein gelten für ein Stringaar $(\varphi, y)$ die Schritte

1. Ein Zeichen von rechts nach links (shift): $(\varphi, \underbrace{d\omega}_{=:y}) \overset{s}{\vdash} (\varphi d, \omega)$
2. Reduzieren am rechten Rand des linken Strings (reduce): $(\underbrace{\psi\beta}_{\varphi}, y) \overset{r}{\vdash} (\psi A, y)$ mit $A \to \beta$.

Starte mit $(\epsilon, x)$, das Ende liegt bei $(S, \epsilon)$.

**Anmerkung**
Das Reduzieren sollte möglichst weit links geschehen. Es entsteht dann eine Rechtsableitung. In Symbolen:

$$S \underset{rm}{\overset{*}{\Rightarrow}} x$$

Die Reduzierung mit $A \to beta$ bei $(\varphi\beta, y)$ ist ok, falls $S' \underset{rm}{\overset{*}{\Rightarrow}} \psi A y \underset{rm}{\Rightarrow} \psi\beta y$.

$\psi\beta$ ist ein brauchbarer String für $A \to \beta$. Was wären Anfänge eines brauchbaren Strings? Ein solcher "brauchbarer Anfang" wird auch viable prefix genannt. .

Die Aufgabe besteht also darin, für jede Produktion $A \to \beta$ die brauchbaren Strings und die brauchbaren Anfänge zu finden. Bzw. wenn $(\varphi, y)$ gegeben ist, ob $\varphi$ ein brauchbarer Anfang für $A$ ist (oder sogar ein brauchbarer String) und für welche Produktionen dies gilt.

Der Anfang eines $\psi\beta$ ist entweder links von $\beta$ (speziell $\epsilon$) oder erfasst den Anfang von $\beta$ (bis hinzu ganz $\beta$).

Produktionen ("item") mit Punkt: $A \to \beta_1 \cdot \beta_2$, $\beta = \beta_1\beta_2$.
Speziell gilt

1. für $A \to \epsilon$ nur 1 item $A \to \cdot$
2. für $A \to \cdot\beta$: brauchbarer Anfang links von $\beta$
3. für $A \to \beta\cdot$: vollständiges item

**Beispiel**

Grammatik $S' \to Sc$, $S \to SA \,|\, A$, $A \to aSb \,|\, ab$.
Finde brauchbare Anfänge für $A \to ab$.
Ein brauchbarer String wäre z.B. $aSab$ wegen

$$S' \overset{*}{\Rightarrow} \underbrace{aS}_{\psi}\, A \underbrace{bc}_{y} \Rightarrow\overset{*}{\Rightarrow} \underbrace{aS}_{\psi}\underbrace{ab}_{\beta}\underbrace{bc}_{y}$$

Ein brauchbarer Anfang ohne $\beta$ Anteil ist $\epsilon, a, aS$, mit $\beta$ Anteil $aSa, aSab$. Zu brauchbaren Anfängen für $A \to \beta$ angeben, welcher Teil von $\beta$ erfasst wird.

$S$ ist brauchbarer Anfang für: $Sc$ ist brauchbarerer String für $S' \to Sc$ und $SA$ ist brauchbarerer String für $S \to SA$.

Mit brauchbaren Anfang sind also erreichbar 2 items $S' \to S \cdot c$ und $S \to S \cdot A$.

---

**Satz 4.20** *Die brauchbaren Anfänge einer Grammatik sind der Sprachschatz eines $\epsilon$NFA $M$. Zustände des $M$ : $q_0$ und alle items.*
*Die Übergänge:*

1. $\delta(q_0, \epsilon) = \{S \to \cdot\beta \,|\, S \to \beta \textit{Produktion}\}$
2. $\delta(A \to \beta_1 \cdot B\beta_2\,,\, \epsilon) = \{B \to \cdot\gamma \,|\, B \to \gamma \textit{Produktion}\}$ ***reduce***
3. $\delta(A \to \beta_1 \cdot B\beta\,,\, B) = \{A \to \beta_1 B \cdot \beta_2\}$ *mit* $B \in V \cup T$. ***shift***

Ein brauchbarere String $\psi\beta$ führt zu einem item $A \to \beta\cdot$. Reduce wird nach Erreichen eines vollständigen items durchgeführt.

**Beispiel**

(Siehe Blatt) $aSa$ ist brauchbarer Anfang.
$a\underline{SA}$ ist brauchbarere String für $I_0 \xrightarrow{sSA} I_5$. ($I_5$ ist vollständiges item)
$\underline{aSb}$ ist brauchbarere String für $I_0 \xrightarrow{aSb} I_8$. ($I_8$ ist vollständiges item)
Diesen Automaten nennt man auch LR0 Automat.

---

Die eindeutige bottom-up Analyse bei $(\varphi, y)$ läuft wie folgt:

- Reduziere, falls $\varphi$ ein brauchbarer String gemäß dem DFA ist, mit der *einen* Produktion.

- Lesen, falls einer der anderen Zustände erreicht wird.

**Definition 4.21** *Eine CFG heißt* ***LR(0) Grammatik*** *genau dann wenn gilt:*

1. *Das Startsymbol bei Produktionen steht nie rechts.*
2. *Jeder Zustand des DFA mit* $A \to \beta\cdot$ *enthält*
    - *kein anderes vollständiges item (kein reduce-reduce Konflikt)*
    - *kein item der Form* $A \to \beta \cdot a$ *mit* $a \in T$ *(kein shift reduce Konflikt)*

### 4.6.1 Syntaktische Analyse für LR(0) Sprachen

- Verarbeite die Eingabe zeichenweise von links nach rechts.
- Beginne bei Zustand $I_0$ (Anfangszustand) des DFAs.
- Notiere auf dem Stack den bisherigen Weg im DFA.
    - Für die Eingabe $x = a_1a_2 \ldots a_i \ldots a_n$ notiere $I_0a_1q_1a_2q_2 \ldots a_iq_i$ mit $\delta(q_{i-1}, a_i) = q_i$ gemäß DFA
- Bei einem *shift* verkürze die Resteingabe.
- Beim Erreichen eines *reduce* Zustandes $A \to \beta\cdot$ reduziere $(\psi\beta, y) \vdash (\psi A, y)$, wobei die Resteingabe bleibt.
- Setze das Verfahren beim mit $\psi$ erreichten Zustand $q\varphi$ Übergang $\delta(q\psi, A)$ gemäß dem DFA fort.
- Das Ende ist erreicht, falls auf dem Stack $I_0S'$ und Resteingabe $\epsilon$ vorliegt.

**Beispiel**

Zur obigen Grammatik unter zu Hilfe name des Automaten.

| *Stack* | *Resteingabe* | *Zustand* | *Aktion* | |
|---|---|---|---|---|
| $0$ | $aababbc$ | $S$ | $0a = 3$ | |
| $0a3$ | $ababbc$ | $S$ | $3a = 3$ | |
| $0a3a3$ | $babbc$ | $S$ | $3b = 7$ | |
| $0a3\underline{a3b7}$ | $abbc$ | $R$ | $A \to ab$ | $3A = 2$ |
| $0a3\underline{A7}$ | $abbc$ | $R$ | $S \to A$ | $3S = 6$ |
| $0a3S6$ | $abbc$ | $S$ | $6a = 3$ | |
| $0a3S6a3$ | $bbc$ | $S$ | $3b = 7$ | |
| $0a3S6\underline{a3b7}$ | $bc$ | $R$ | $A \to ab$ | $6A = 5$ |
| $0a3\underline{S6A5}$ | $bc$ | $R$ | $S \to SA$ | $3S = 6$ |
| $0a3S6$ | $bc$ | $S$ | $6b = 8$ | |
| $0\underline{a3S6b8}$ | $c$ | $R$ | $A \to aSb$ | $0A = 2$ |
| $0A2$ | $c$ | $R$ | $S \to A$ | $0S = 1$ |
| $0S1$ | $c$ | $S$ | $1c = 4$ | |
| $0\underline{S1c4}$ | $\epsilon$ | $R$ | $S' \to Sc$ | $0S'$ |
| $0S'$ | | | | |

# Index