Mitschrift
Formale Sprachen und Automaten

Prof. Braun
Sommersemester 2002
LaTeXed by Leo Bär

Inhaltsverzeichnis

1 Varianten endlicher Automaten **1**

1.1 Der 2-Wege-Automat 2

1.2 Automaten mit Ausgabe 5

2 Schaltwerke **9**

2.1 Schaltnetze und Schaltwerke 9

2.2 Schaltwerke für Polynome 11

3 Inhärente Mehrdeutigkeit bei kontextfreien Sprachen **19**

4 Syntaktische Analyse kontextfreier Sprachen **23**

4.1 Lexikalische Analyse 23

4.2 Syntaktische Analyse (Grundlagen) 24

4.3 Sackgassenfreie Analyse durch Links- und Rechts-kontext 26

4.4 Operator- und Präzedenzsprachen 29

4.5 LL1 Sprachen 31

4.6 LR Sprachen 34

5 Deterministische Kellerautomaten **43**

6 Transition Networks **49**

6.1 Selbsteinbettung 49

6.2 Basic Transition Networks (BTN) 51

7 Das λ Kalkül **53**

7.1 Vorbemerkungen 53

7.2 Formales λ Kalkül 53

7.3 Normalformen und Konfluenz 56

7.4 Kombinatoren 56

8 Turingmaschienen und Unentscheidbarkeit **59**

8.1 Das Post'sche Korrespondenzproblem (PKP) 60

Vorwort

Diese Mitschrift basiert auf meinen Unterlagen und Kopien der Unterlagen meiner Kommilitonen. Es kann daher keinerlei Garantie für 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. Vor allem die etwas länger geratenen Beweise sollten kritisch betrachtet werden, da diese oft nur mitgeschrieben und noch nicht überarbeitet worden sind.

Desweiteren ist die Darstellung von terminal und nonterminal Zeichen nicht einheitlich. Oft existieren auch noch Schreibfehler, welche zu Verwechslungen führen können (z.B. ω statt w und umgekehrt). Wer Schreibfehler findet, Unklarheiten entdeckt, oder sonstige Kritik hat, kann mir diese unter `Leo.Baer@gmx.de` mitteilen.

Leo Bär 15. August 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
- Σ Alphabet
- δ 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

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

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

Nichtdeterministische Automaten

Es gilt

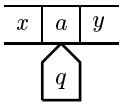
$$\delta(q, a) \subseteq Q \qquad \hat{\delta}(q, wa) = \bigcup_{r \in \hat{\delta}(q,w)} \hat{\delta}(r, a) \qquad T(M) = \big\{w \in \Sigma^* \big| \hat{\delta}(q_0, w) \cap F \neq \emptyset\big\}$$

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 α, β *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

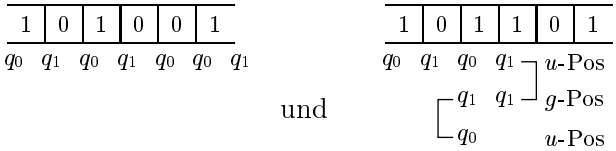
$$q_0\,101001 \vdash \qquad 1\,q_1\,01001 \vdash \qquad 10\,q_0\,1001 \vdash \qquad 101\,q_1\,001 \vdash$$
$$1010\,q_0\,01 \vdash \qquad 10100\,q_0\,1 \vdash \qquad 101001\,q_1\, \vdash \qquad \text{akzeptiert}$$

Aber

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

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

Im Beispiel trat 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:



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.
Im 1. Fall wäre diese Teil CS ungültig, im 2. Fall ist q unter den Zuständen und die gesamte CS ungültig. ∎

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

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

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
   $q_0' := [q_0] \qquad F' := \big\{[p] \big| p \in F\big\} \qquad \delta'(c, a) = \big\{d \in Q' \big| c \varrho d \text{ über } a\big\}$
2. Zeige $w \in T(M) \Longrightarrow w \in T(M')$.
   Sei $w = a_1 a_2 \dots a_n$, $c_0, c_1, \dots, c_n$ mit $c_0 = [q_O]$ und $c_n = [p]$.
   Zu zeigen also $c_{i-1} \varrho c$ über a_i und $c_{i-1} = [q_1, \dots, q_k], c_i = [p_1, \dots, p_l]$. 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
   $[q_3 \dots q_k] \varrho [p_1 \dots p_l]$ über a_i oder
   $[q_2 \dots q_k] \lambda [p_2 \dots p_l]$ 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 $w = a_1 \dots a_n \in T(M')$, 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 $w_i := a_1 \dots a_i$; $\quad w_0 := \epsilon$; $\quad c_o \underset{M'}{\overset{a_1 \dots a_i}{\vdash}} c_i$ und $c_i = [p_1 \dots p_l]$ 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 $j \in \{2, 4, \dots, l_1\}$ 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 $[q_0] \underset{M'}{\overset{\epsilon}{\vdash}} [q_0]$, also (3a) erfüllt und (3b) leer.

   Induktionsvorraussetzung
   (3a) und (3b) gelten für i_1. Genauer:
   $c_0 \underset{M'}{\overset{a_i \dots a_{i-1}}{\vdash}} c_{i-1}$, $c_{i-1} = [q_1, \dots, q_k]$ mit k ungerade.
   Dann gilt
   (a) M verlässt a_{i-1} zum ersten Mal nach rechts in q_1, also $q a_{i-1} q_1 R$.
   (b) Für $j \in \{2, 4, \dots, k-1\}$ 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 $c_{i-1} = [q_1, \dots, q_k]$, $c_i = [p_1, \dots, p_l]$ und $c_{i-1} \varrho c_i$ über a_i. Zu (3a).
   a_i erstesmal gelesen beim ersten Verlassen von $a_1 \dots a_{i-1}$ nach rechts.
   Nach IV a_i erstmalig gelesen in q_1. Nach Regel $q_1 a_i \dots$ 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) $q_1 a_i q_2 L [q_3 \dots q_k] \varrho [p_1 \dots p_l]$
   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

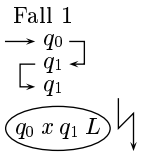
verlässt M danach a_{i-1} erstmalig nach rechts in q_3. Es gilt also dasselbe wie zuvor, aber mit echt verkürztem $c'_{i_1} = [q_3 \dots q_k]$. Bei Fortsetzung tritt also entweder Fall 1 ein oder weitere Verkürzungen des c_{i-1}. Spätestens bei der Verkürzung des c_{i-1} auf $[q_k]$ entsteht Fall 1.

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:

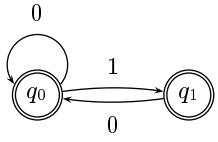


Fall 2 würde den Fall $q_1\,1\,q_1\,L$ oder $q_1\,0\,q_2\,R$ benötigen, der existiert. Um aber in diesen Zustand zu kommen, würde eine Regel $q_?\,0 q_1 R$ benötigt werden, die nicht vorhanden ist.

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)

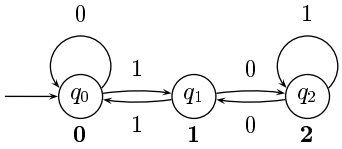
1. *$M = (Q, \Sigma, \mathbf{\Delta}, \delta, \lambda, q_0)$ mit $\delta : Q \to \Delta$*
2. *$w = a_1 a_2 \dots a_n \in \Sigma^*$ und $q_o \overset{a_1}{\vdash} q_1 \overset{a_2}{\vdash} q_2 \vdash \dots \overset{a_n}{\vdash} q_n$*
   Mit der Ausgabe $x = \lambda(w) := \lambda(q_0)\lambda(q_1)\dots\lambda(q_n)$.
   Weiterhin gilt $\lambda(\epsilon) := q_0$

Beispiel:

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

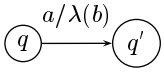
i mod 3	I	$i0$	$i0$ mod 3	$i1$	$i1$ mod 3
0	$3m$	$6m$	0	$6m+1$	1
1	$3m+1$	$6m+2$	2	$6m+3$	0
2	$3m+2$	$6m+4$	1	$6m+5$	2

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



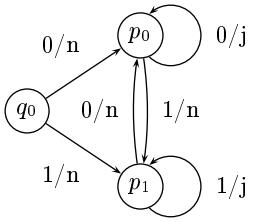
Definition 1.11 (Mealy Automat)

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* $M' = (Q', \Sigma, \Delta, \delta', \lambda', q_0')$

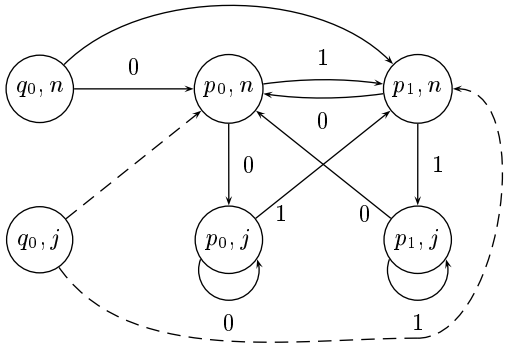
Beweis
Die Idee des Beweises besteht darin, das beim Mealy Übergang ausgegebene Zeichen zum Folgezustand hinzuzunehmen. Man erhält also einen M'-Zustand $[q, b]$ mit $q \in Q$, $b \in \Delta$, $Q' := Q \times \Delta$, und die Abbildung $\delta'\big([q, b], a\big) = \big[\delta(q, a), \lambda(q, a)\big]$ (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



Kapitel 2

Schaltwerke

2.1 Schaltnetze und Schaltwerke

Boole'sche Funktion $y = f(x_1, \ldots, x_n)$ 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:

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

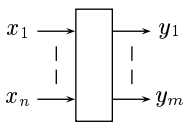
Beispiel:

---

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

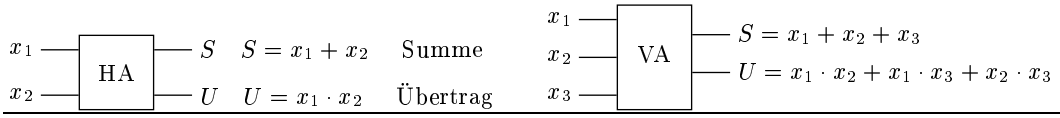
also $f(x_1, x_2, x_3) = (x_1 \wedge \bar{x}_2 \wedge \bar{x}_3) \vee (\bar{x}_1 \wedge x_2 \wedge \bar{x}_3) \vee (x_1 \wedge \bar{x}_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge x_3)$

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



Beispiel:

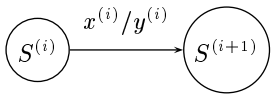
Halbaddierer und Volladdierer



Definition 2.2 (Schaltwerk)
*Ein **Schaltwerk** ist ein Schaltnetz mit Speicher. Ein **Speicher** besteht aus Speichereinheiten zu je 1 Bit. Diese besitzen die Werte* $s_0, s_1, \ldots, s_{r-1}$.

Die Ausgabe eines Schaltwerks ist Takt gesteuert und geschieht zu den Zeitpunkten $t_0, t_1, \ldots, t_n$. *Die Eingabe ist definiert als* $x^{(i)} := (x_1, \ldots, x_n)^{(i)}$. *Die Ausgabe* $y^{(i)} := (y_1, \ldots, y_m)^{(i)}$ *ist abhängig von* $x^{(i)}$ *und dem Zustand* $s^{(i)} := (s_0, \ldots, s_{r-1})^{(i)}$ *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.

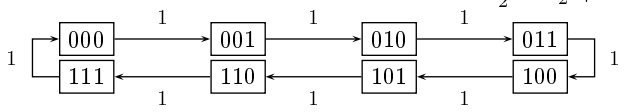
Beispiel:

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 $000, 001, \ldots, 111$ durchlaufen werden.
Es gilt $(s, e) \vdash s'$, $s = s_0 s_1 s_2$, $s' = s'_0 s'_1 s'_2$
Dann

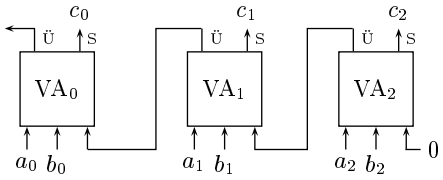
$$s'_0 = s_0 + e s_1 s_2 \qquad s'_1 = s_1 + e s_2 \qquad s'_2 = s_2 + e$$

Also

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

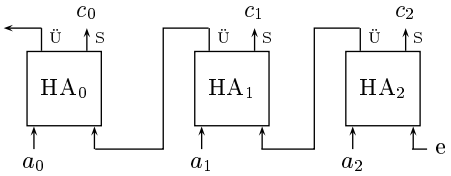


Veranschaulichung für die Übergangsregel des D_3. Start bei 000, mehrmaliges addieren von 001. Addiere $a_0a_1a_2 + b_0b_1b_2 = c_0c_1c_2$, also



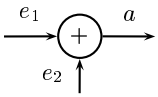
Für $b_0b_1b_2 = 00e$ ergibt sich eine Folge von Halbaddieren:

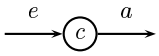
$S_j = a_j \quad S'_j = c_j$
$c_2 = a_2 + e \quad u_2 = a_2 \cdot e$
$c_1 = u_2 + a_1 = a_1 + a_2e \quad u_1$
$u_1 = u_2 \cdot a_1 = a_2a_1e$
$c_0 = u_1 + a_0 = a_2a_1e + a_0$

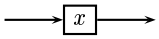


2.2 Schaltwerke für Polynome

Sei $a(x) = a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1}$ 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 $x \in \{0, 1\}$ 

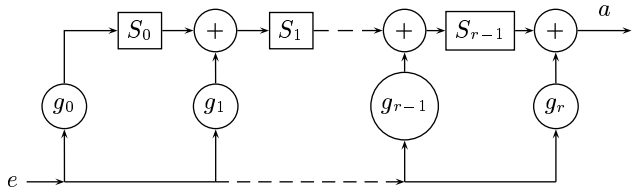
Bei 1. Takt geschieht also folgendes:

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

Ein Polynom wird dann als $s(x) = s_0 + s_1x + \cdots + s_{r-1}x^{r-1}$ 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:

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

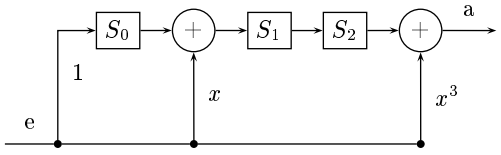
■

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 $b_k, \ldots, b_0, \underbrace{0, \ldots, 0}_{r-mal}$ *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 $a_j = c_{k+r-j}$. ■

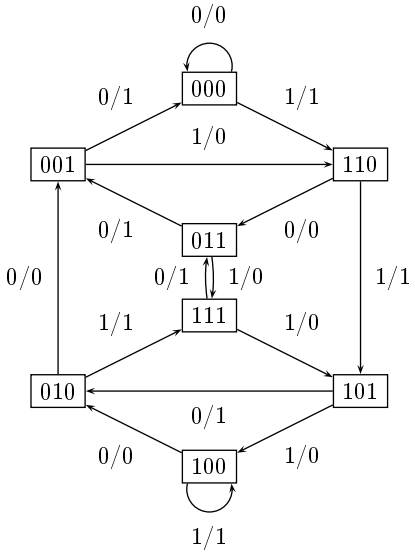
Beispiel:

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



Es gilt also $S_0' = e$, $S_1' = e + S_0$, $S_2' = S_1$, $a = e + S_2$, 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:

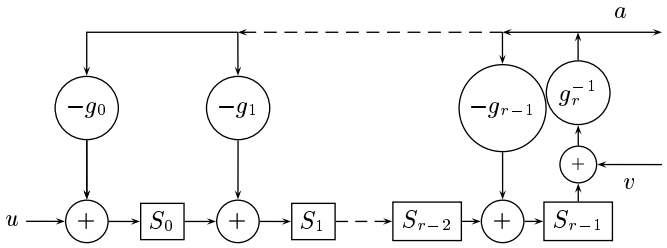


1. $b(x) = 1 + x^2 + x^3$, $g(x) = 1 + x + x^3$
   Eingabe: 1101:000 ($r = 3$ in $g(x)$, also 3 Nullen anhängen.)
   Ausgabe: 1111111
   Also $c(x) = x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$
   Beachte auch den Zusammenhang zu den irredzuieblen Faktoren von $x^7 + 1 = (x+1)(x^3 + x^2 + 1)(x^3 + x + 1)$ (Codierungstheorie).

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, ersetze s durch S]
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

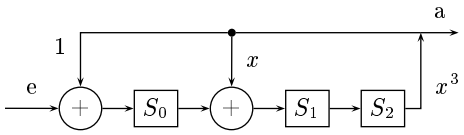
$$\begin{aligned} s^{(i+1)}(x) &= \big(x \cdot s^{(i)}(x) + d_{n-i-1}\big) \bmod g(x) \\ &= \big(x \cdot \big(d^{(i-1)}(x) \bmod g(x)\big) + d_{n-i-1}\big) \bmod g(x) \\ &= \ldots \\ &= d^{(i)}(x) \bmod g(x) \end{aligned}$$

3. $s^{(n)}(x) = \underbrace{d^{(n-1)}(x)}_{d(x)} \bmod g(x)$

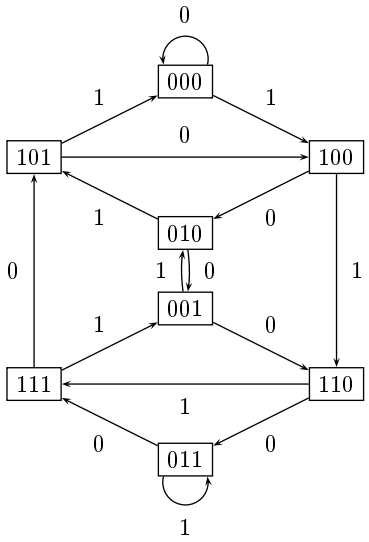
∎

Beispiel:

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



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



1. $d(x) = x^7 + 1$ Eingabe 10000001 liefert Ergebnis 000, also $g(x) \mid x^7 + 1$
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 $g(x) \mid \underbrace{x^6 + x^2 + 1}_{=(g(x))^2}$
4. $d(x) = x^4 + x^3 + x^2 + 1$ Eingabe 11101 liefert Ergebnis 000, also $g(x) \mid d(x)$
5. $d(x) = g(x)$ Eingabe 1011 liefert Ergebnis 000, also $g(x) \mid g(x)$

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 $L_0 = \{a^i b^i c^j | i,j \geq 1\} \cup \{a^i b^j c^j | i,j \geq 1\}$ *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

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

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]

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
   $S \overset{*}{\Rightarrow} w_1 A w_2 \overset{*}{\Rightarrow} w_1 b^p A c^p w_2 \overset{*}{\Rightarrow} w_1 b^p w c^p w_2 =: z'$, 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 $S \overset{*}{\Rightarrow} w_1 A w_2 \overset{*}{\Rightarrow} w_1 w w_2 = z_1$ mit $A \overset{*}{\Rightarrow} a^n A b^n$, dann
   $S \overset{*}{\Rightarrow} w_1 A w_2 \overset{*}{\Rightarrow} w_1 a^p A a b^p w_2 \overset{*}{\Rightarrow} w_1 a^p w b^p w_2 =: z_0$, mit $|z_0|_a = 2p$, $|z_0|_b = 2p$, $|z_0|_c = 2p$ und wegen $S \overset{*}{\Rightarrow} z_0$ 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
$S \overset{*}{\Rightarrow} w_1 A w_2 \overset{*}{\Rightarrow} w_1 w w_2 = a^r b^s c^t$ mit $r = s \vee s = t$, dann gilt auch $S \overset{*}{\Rightarrow} w_1 x^q A y^q w_2 \overset{*}{\Rightarrow} w_1 x^q w y^q w_2$. 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
   $x = b^i, y = b^j \Longrightarrow S \overset{*}{\Rightarrow} w_1 b^{iq} w b^{jq} w_2 =: z_1$,
   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**

[1] Hier ist die Struktur und der Zusammenhang noch nicht ganz perfekt

$y \neq c^n$

Beweis

Sei $y = c^n$ mit $n \geq 1$, dann gilt:

$S \overset{*}{\Rightarrow} w_1 a^{mq} w c^{nq} w_2 =: z_2$ (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.*

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.

Beispiel:

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

Beispiel:

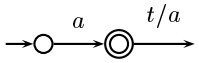
Eine Zahl in PASCAL (ohne Vorzeichen) besteht aus $z^+[.z^+][E[+|-]z^+]$ mit z Ziffer.

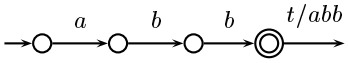
Eventuell bietet sich aber eine einfachere Vorgehensweise bei mehreren Varianten an. Konstruiere zuerst für jede Variante einen DFA[1], fasse diesen zu einem ϵDFA zusammen und wandle diesen in einen DFA um.

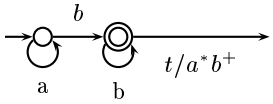
[1] deterministic finite automaton?

Beispiel:

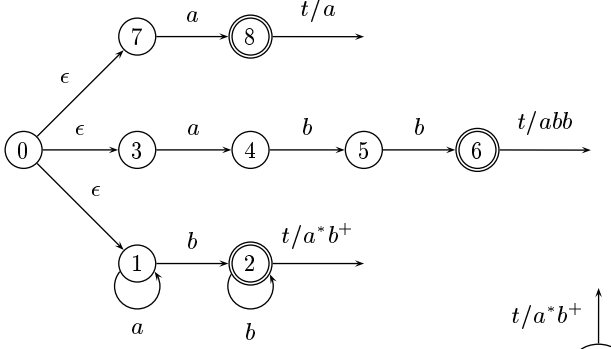
3 reguläre Ausdrücke mit $\Sigma = \{a, b\}$:

1. a 

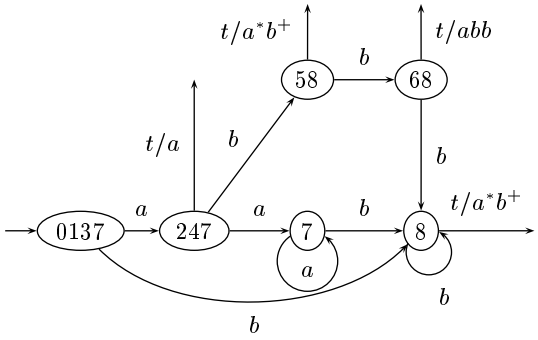
2. abb 

3. a^*b^+ 

Zusammengefasst zu einem ϵDFA 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)[2]

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, β), wobei y der Rest auf dem Leseband und β 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]Ein **Parser** ist ein Verfahren zur Lösung dieses Problems

2. Über Endzustand
   $(q_0, w, z_0) \vdash (q_f, \epsilon, \beta)$ mit $q_f \in F$ und β 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 Σ, $V \cup T$ dem Kelleralphabet Γ und S dem ? z_0 entspricht).

Es gilt:

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

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, ϵ, S) $\quad (q_0, bDAe)$
6. (q_0, ϵ, D) $\quad (q_0, dD), (q_0, d)$
7. (q_0, ϵ, A) $\quad (q_0, aA), (q_0, a)$

oder

$bddae$	S
$\not b ddae$	$\not b DAe$
$ddae$	$DAe \vdash ddae, dAe \vdash \ldots$ ↯
$\not d ddae$	$\not d DAe$
dae y	$DAe \vdash \ldots$ ↯
$\not d ae$	$\not d Ae$
ae	$Ae \vdash \ldots$ ↯
$\not a e$	$\not a e$
$\not e$	$\not e$
ϵ	ϵ (akzeptiert)

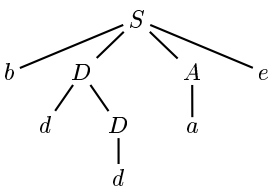
Wobei die Fälle 1 bis 4 (q_0, ϵ) entsprechen.

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 ↯ oder ae , $dDae$ ↯
- auch möglich (von ae, Ae)
  ae , $Ae \vdash ae$, $aAe \vdash e$, Ae
  $\vdash$ entweder e , ae ↯ oder e , aAe ↯

4.2.2 Zweite Technik: Bottom-up-Analyse

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

Beispiel:

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

$$
\begin{array}{lll}
x = \underline{a} + a * a + a & \vdash & \\
\underline{F} + a * a + a & \vdash & \\
\underline{T} + a * a + a & \vdash & \\
E + \underline{a} * a + a & \vdash & \\
E + \underline{F} * a + a & \vdash & \\
E + T * \underline{a} + a & & \vdash E * a + a \quad \vdash \ldots \quad \lightning \\
E + \underline{T * F} + a & & \vdash E * F + a \quad \vdash \ldots \quad \lightning \\
\underline{E + T} * a & \vdash & \\
E + a & \vdash & \\
E + F & \vdash & \\
E + T & \vdash & \\
E & &
\end{array}
$$

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.*

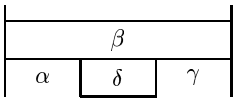
Definition 4.5 *$\beta \in (V \cup T)^*$ heißt genau dann **lokale Mehrdeutigkeit**, wenn für $\beta = \alpha\delta\gamma$ entweder*

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

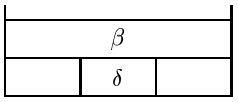
oder 2. $A \to \alpha\gamma$, $B \to \delta\gamma$ mit $\gamma \neq \epsilon$

gilt.

1.

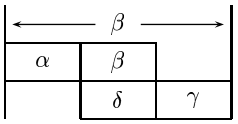


$E \to \underbrace{E + T}_{\beta}$ $\qquad E \to \underset{\uparrow \delta}{T}$



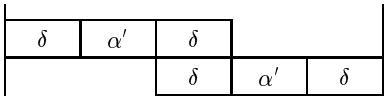
$A + B$

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 λ zunächst erhalten und wird beidseitig nur durch $\varrho \vdash A$ ersetzt.
   Sei $\varrho \subseteq \varphi_i$ für $i = r, \ldots, n$ mit $\varrho \not\subseteq \varphi_{r-1}$.
   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 α_1. bzw. Anfang des α_2.
Z.B. $B \to \beta_1 CAD\beta_2$, dann ist das erste Zeichen des α_2 der Anfang eines aus D erzeugten Strings.

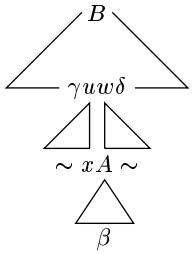
Das letzte Zeichen des Strings des α_1 ist das Ende eines aus C erzeugten Strings, d.h. $\beta \vdash A$ nur möglich in gewissen "Kontexten".

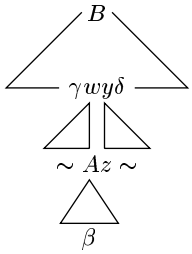
Definition 4.7 *$x \in V \cup T$, dann*

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

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_{arith}, $E \to E + T$, $E \to T$, $T \to T * F$, $T \to F$, $F \to (E)$, $F \to a$.
Lokale Mehrdeutigkeiten

$$\underbrace{\overset{E}{T} * \overset{T}{F}}_{T} \qquad \underbrace{E + \overset{E}{T}}_{E} \qquad \underbrace{E + \overset{\overset{T}{\overline{T * F}}}{T}}_{E}$$

Beispiel: Anmerkung

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

1. Berechne α- und w-Klassen

	α	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 β ein Zeichen aus L_T steht. Z.B.: $T * F \vdash T * T$

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

$\beta \vdash T$ 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) *γ ist genau dann eine **Phrase**, wenn $S \overset{*}{\Rightarrow} \alpha A \beta$ und $A \overset{*}{\Rightarrow} \gamma$.*

γ ist eine Primphase, gdw. γ eine Phrase ist, in γ terminale Symbole [sind?], und keine echte Teilphrase von γ terminale Symbole enthält.

Beispiel:

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

γ_1 Phrase: $E \overset{*}{\Rightarrow} \gamma_1$, Primphrase: Teilphrase nur F (denn $E \overset{*}{\Rightarrow} F$).
γ_2 Phrase: $E \Rightarrow T \overset{*}{\Rightarrow} \gamma_2$, keine Primphrase.

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

1. $a \doteq b \iff A \to \alpha ab\gamma$ *oder* $A \to \alpha aBb\beta$
2. $a \lessdot \iff A \to \alpha aB\beta$ *und* $B \overset{*}{\Rightarrow} b\gamma$ *oder* $B \overset{*}{\Rightarrow} Cb\gamma$
   D.h. beim erzeugen entsteht a vor b
3. $a \gtrdot b \iff A \to \alpha Bb\beta$ *und* $B \overset{*}{\Rightarrow} \gamma a$ *oder* $B \overset{*}{\Rightarrow} \gamma aC$
   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 $S \overset{*}{\Rightarrow} \alpha ab\beta$ oder $S \overset{*}{\Rightarrow} \alpha aBb\beta$ 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 $S \overset{*}{\Rightarrow} \pi ab\mu$ oder $S \overset{*}{\Rightarrow} \pi aDb\mu$ stets wenigstens eines aus $\doteq, \lessdot, \gtrdot$*

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

Fall 1
$S \overset{*}{\Rightarrow} \pi aC\mu' =: \delta$, $C \overset{*}{\Rightarrow} b\delta$ oder $C \overset{*}{\Rightarrow} Db\delta$.
Es existiert eine Produktion $A \to \alpha aB\beta$ 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)
$S \overset{*}{\Rightarrow} \lambda' Cb\mu$, $C \overset{*}{\Rightarrow} \delta a$ bzw. $C \overset{*}{\Rightarrow} \delta aD$
also $a \gtrdot b$

■

Lemma 4.13 *In einer Präzedenzsprache gilt für $a, b \in T$ mit $S \overset{*}{\Rightarrow} \pi ab\mu$ bzw. $S \overset{*}{\Rightarrow} \pi aDB\mu$ 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:

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

Beispiel:

G_{arith}; $\#a + a * (a + a)\#$
$\#\overset{\lessdot}{[}a\overset{\gtrdot}{]} + \overset{\lessdot}{[}a\overset{\gtrdot}{]} * \overset{\lessdot}{[}\overset{\lessdot}{(}a\overset{\gtrdot}{]} + \overset{\lessdot}{[}a\overset{\gtrdot}{]}\overset{\gtrdot}{)}]\#$
$\#\overset{\lessdot}{[}F + \overset{\lessdot}{[}F * \overset{\lessdot}{[}\overset{\lessdot}{(}[F + F]\overset{\gtrdot}{)}\overset{\gtrdot}{]}\#$
$\#\overset{\lessdot}{[}F + \overset{\lessdot}{[}F * \overset{\lessdot}{[}\overset{\doteq}{(}E\overset{\gtrdot}{)}\overset{\gtrdot}{]}\#$
$\#\overset{\lessdot}{[}F + \overset{\lessdot}{[}F + F\overset{\gtrdot}{]}\#$
$\#\overset{\lessdot}{[}F + T\overset{\gtrdot}{]}\#$
$\#E\#$

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 φ 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, $S \to bDAe$, $D \to dD \mid \epsilon$ und $A \to aA \mid \epsilon$. 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* ϵ *Regeln.*

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

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

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 $n \geq 1$.
follow(S) : $S \in \text{last}(S)$, S steht nie rechts.
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. $D \to dD\underset{\varphi}{\underset{\uparrow}{\epsilon}}$ $\quad \text{first}(\varphi) = \emptyset$

follow(A) : $A \in \text{last}(A)$, A steht rechts bei:

1. $S \to bDA\underset{\varphi}{\underset{\uparrow}{e}}$ $\quad \text{first}(\varphi) = \{e\}$ $\qquad$ 2. $A \to aA\underset{\varphi}{\underset{\uparrow}{\epsilon}}$ $\quad \text{first}(\varphi) = \emptyset$

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*

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

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

Beispiel:

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

Top-down Analyse mit Vorschau

	S
$\not b ddae$	$\not b DAe$
$\not d dae$	DAe
	$\not d DAe$
$\not a ae$	DAe
	$\not a DAe$
$\not a e$	DAe
	Ae
	$\not a e$

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:

Prozeduren für obige Beispiele. (read nimmt immer das nächste Zeichen)

```
proc S
  {
    if (z = "b") then read(z) else Fehler;
    call D;
    call A;
    if (z = "e") then read(z) else Fehler;
  }

proc D
  {
    if (z in ("a","e")) then break  /* D -> epsilon */
    else
      {
        if (z = "d") then read(z) else Fehler;
        call D; /* D -> dD */
      }
  }

proc A
  {
    if (z = "e") then break  /* A -> epsilon */
    else
      {
        if (z = "a") then read(z) else Fehler;
        call A; /* A -> aA */
      }
  }
```

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 Stringpaar (φ, 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 (ϵ, x), das Ende liegt bei (S, ϵ).

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 (brS) für $A \to \beta$. Was wären Anfänge eines brauchbaren Strings? Ein solcher "brauchbarer Anfang" (brA) 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 (φ, y) gegeben ist, ob φ 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 β (speziell ϵ) oder erfasst den Anfang von β (bis hinzu ganz β).

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 β
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

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

Ein brauchbarer Anfang ohne β Anteil ist ϵ, a, aS, mit β Anteil $aSa, aSab$. Zu brauchbaren Anfängen für $A \to \beta$ angeben, welcher Teil von β 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 ϵNFA[3] M. Zustände des M : q_0 und alle items.*
Die Übergänge:

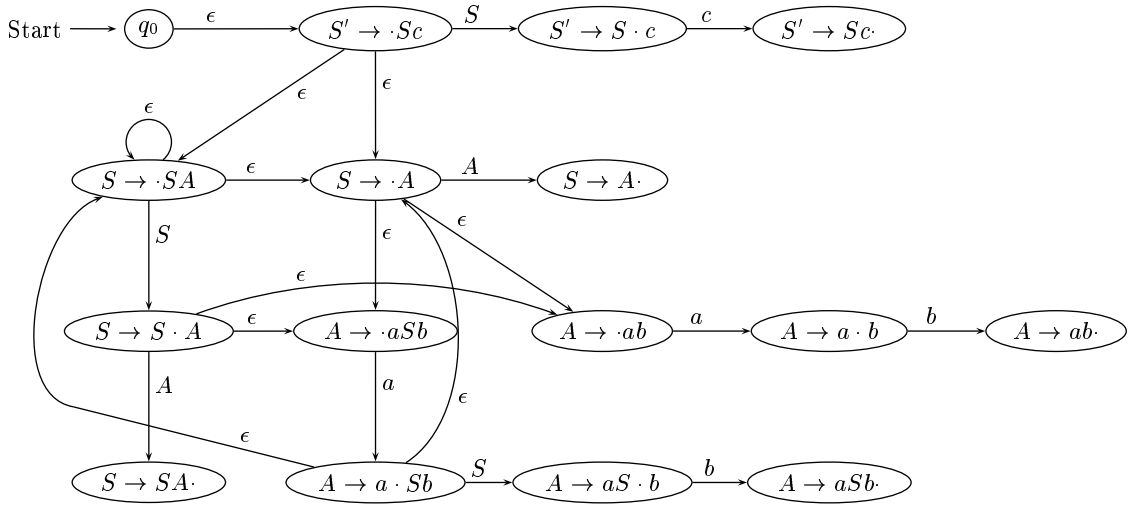
1. $\delta(q_0, \epsilon) = \left\{ S \to \cdot\beta \;\middle|\; S \to \beta Produktion \right\}$
2. $\delta(A \to \beta_1 \cdot B\beta_2\,,\, \epsilon) = \left\{ B \to \cdot\gamma \;\middle|\; B \to \gamma Produktion \right\}$ ***reduce***
3. $\delta(A \to \beta_1 \cdot B\beta\,,\, B) = \left\{ A \to \beta_1 B \cdot \beta_2 \right\}$ *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.

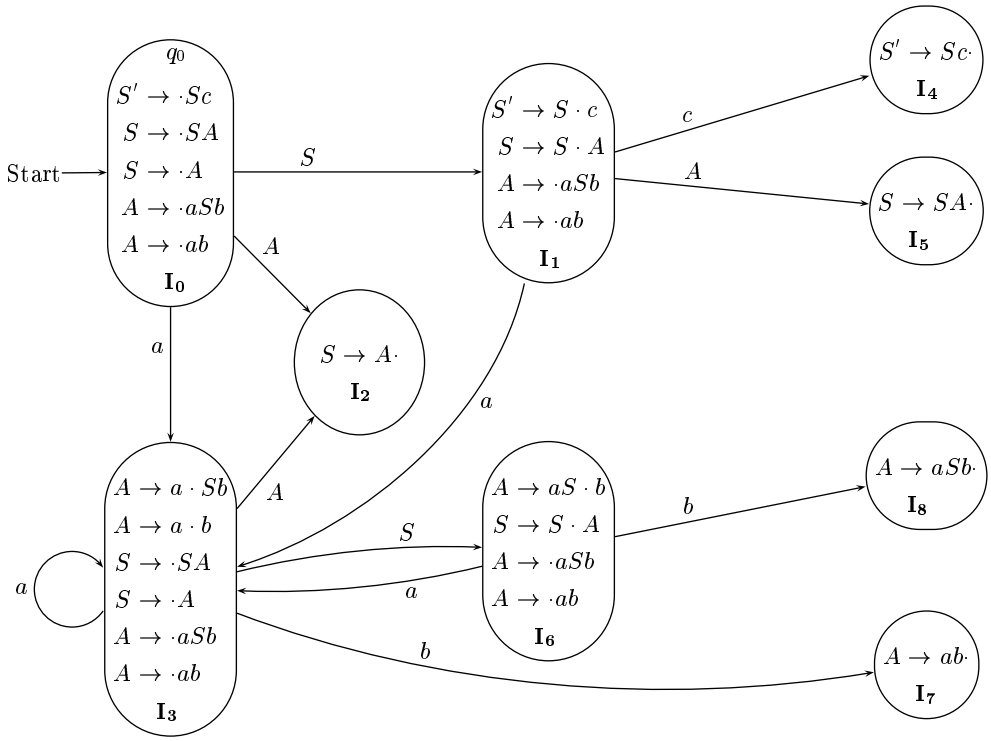
[3] nondeterministic finite automaton?

Beispiel:

Für die obige Grammatik ergibt sich folgender NEA



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)
Aus obigen NEA kann folgender DEA abgeleitet werden:



Die eindeutige bottom-up Analyse bei (φ, y) läuft wie folgt:

- Reduziere, falls φ 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 $x = a_1 a_2 \dots a_i \dots a_n$ notiere $I_0 a_1 q_1 a_2 q_2 \dots a_i q_i$ 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 ψ erreichten Zustand $q\varphi$ Übergang $\delta(q\psi, A)$ gemäß dem DFA fort.
- Das Ende ist erreicht, falls auf dem Stack $I_0 S'$ und Resteingabe ϵ vorliegt.

Beispiel:

Zur obigen Grammatik unter zu Hilfe name des Automaten.

Keller	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	$1c = 4$	
$0\underline{S1c4}$	ϵ	R	$S' \to Sc$	$0S'$
$0S'$				

Definition 4.22 *Eine* ***Parsing Tabelle*** *besitzt*

- *für jeden Shiftzustand S den Folgezustand (terminal / nichtterminal).*
- *Für jeden Reduce-Zustand R die Produktionen und gegebenenfalls "acc" für akzeptierenden R-Zustand.*

Sie ist meist leichter zu lesen als obiger Automat

Beispiel:

Für obigen Automaten ergibt sich folgende Parsing Tabelle für LR(0):
(Grammatik $S' \to Sc$, $S \to SA \mid A$, $A \to aSb \mid ab$.)

	a	b	c	A	S
0	$S3$	–	–	2	1
1	$S3$	–	$S4$	5	–
2	$S \to A$	$S \to A$	$S \to A$	–	–
3	$S3$	$S7$	–	2	6
4		acc		–	–
5	$S \to SA$	$S \to SA$	$S \to SA$	–	–
6	$S3$	$S8$	–	5	–
7	$A \to ab$	$A \to ab$	$A \to ab$	–	–
8	$A \to aSb$	$A \to aSb$	$A \to aSb$	–	–

Definition 4.23 *Eine CFG heist* ***deterministisch (DCFL)*** *gdw.* $L_0 = T(M)$ *und M ein DPDA ist.*

$$\{DCFL\} \subsetneqq \{CFL\}$$

Es gilt: $L_0 \subsetneqq T^*$ ist genau dann eine LR(0) Sprache, wenn L_0 eine DCFL ist und L_0 die "Präfixeigenschaft" besitzt.

Definition 4.24 ***Präfixeigenschaft*** *von* L_0 *gdw. für alle* $x \in L_0$ *folgt, dass kein* $w \in L_0$[4] *mit* $x = wz$ *mit* $z \neq \epsilon$ *existiert*[4].

Die LR(1) sind spezielle CFG für alle DCFLs !?

Definition 4.25 ***LR(1)-item***

1. *Ein Ableitung* $A \to \beta_1 \cdot \beta_2[a]$ *mit* $a \in T$ *heißt LR(1)-item gdw.* $Srm\psi Ayrm\psi\beta y$ *mit* $y = az$.[5]
2. *Eine Ableitung* $A \to \beta_1 \cdot \beta_2[a]$ *heißt* ***gültig*** *für jeden brauchbaren Anfang des brauchbaren Strings* $\psi\beta$

Mehrere LR(1)-items existieren nur mit verschiedenen Vorzeichen: $A \to \beta_1 \cdot \beta_2[a_1, \ldots, a_n]$

[4] Wenn $x \in L_0$, dann existiert kein echter Anfang von x, der ebenfalls in L_0 liegt

[5] denn evtl. $z = \epsilon$ bzw. bei $y = \epsilon \Longrightarrow a := \$$

Definition 4.26 *Ein nicht deterministischer LR(1) Automat $Q : q_0$, LR(1)-Items [besitzt?]*

1. $\delta(q_0, \epsilon) := \{S \to \cdot\beta[\$] \,|\, S \to \beta\}$
2. *(reduce)*
   $\delta(A \to \beta_1 \cdot B\beta_2[a_1, \ldots, a_n], \epsilon) := \{B \to \cdot\gamma[b_1, \ldots, b_m]\}$ *mit* $b_i \in \text{first}(\beta_2)$, *sowie bei* $\beta_2 \stackrel{*}{\Rightarrow} \epsilon$, *auch* $\{a_1, \ldots, a_n\}$
3. *(shift)*
   $\delta(A \to \beta_1 \cdot B\beta_2[a_1, \ldots, a_n], \beta) := \{A \to \beta_1 B \cdot \beta_2[a_1, \ldots, a_m]\}$

Beispiel:

Grammatik $S \to A$, $A \to BA \,|\, \epsilon$, $B \to aB \,|\, b$, also $L(G) = (a^*b)^* = \{ab, abb, \ldots\}$. Dies ist keine LR(0), da die Präfixeigenschaft nicht gilt.

Definition 4.27 (LR(1)) *Eine kontextfreie Grammatik G ist vom Typ LR(1) wenn gilt*

1. *S steht nie rechts.*
2. *Für den DFA zu G gilt:*
   Wenn in einem Zustand ein vollständiges item $A \to \beta \cdot [a_1, \ldots, a_n]$ vorkommt, dann
   (a) *existiert bei diesem Zustand kein item der Form $B \to \beta_1 \cdot a_i\beta_2[a_1, \ldots, a_n]$. Es tritt also keine shift/reduce Kollision auf.*
   (b) *gilt für jedes andere vollständige item $B \to \gamma \cdot [b_1, \ldots, b_m]$ dieses Zustandes: $\{a_1, \ldots, a_n\} \cap \{b_1, \ldots, b_m\} = \emptyset$. Es entsteht also keine reduce/reduce Kollision.*

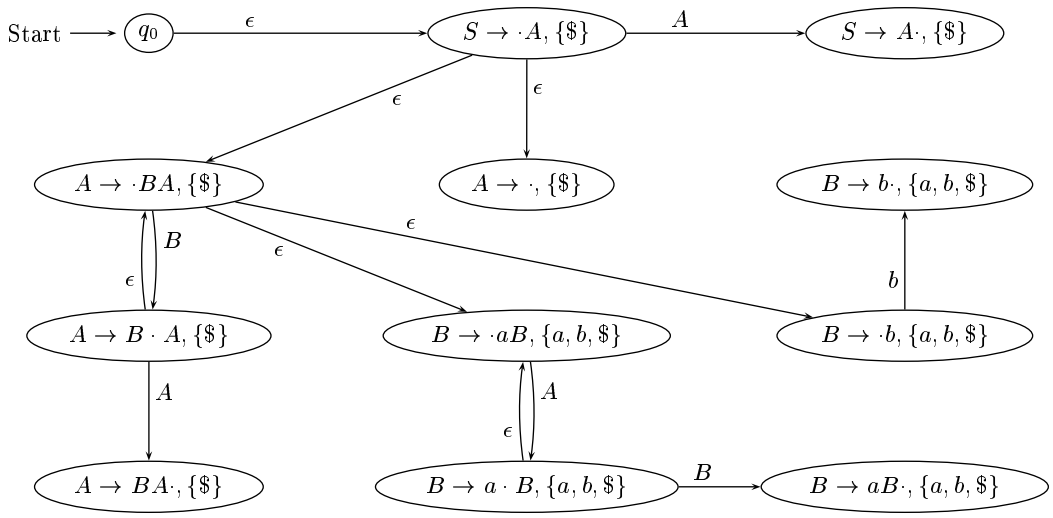
4.6.2 Analyse von LR(1) Sprachen

Diese geschieht analog zur LR(0) Analyse. In jedem DFA Zustand gilt

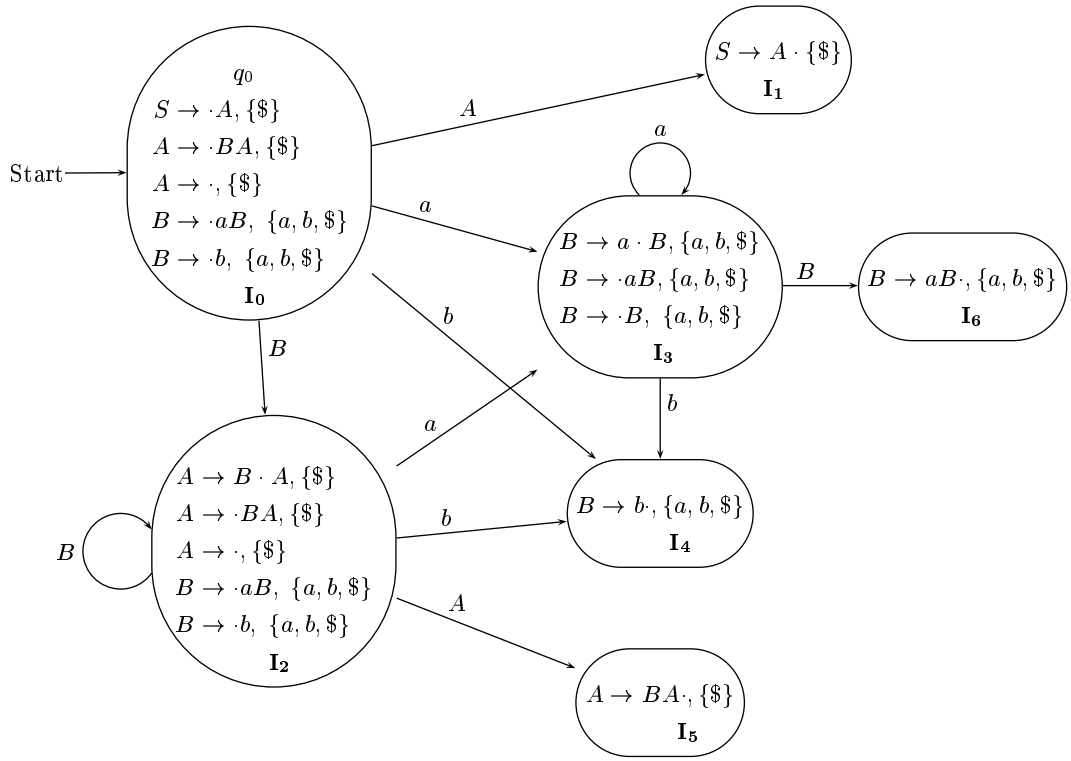
1. $A \to \beta \cdot [a_1, \ldots, a_n]$ mit $A \neq S$ wird ein reduce Schritt mit $A \to \beta$ durchgeführt, falls das Vorschauzeichen $\in \{a_1, \ldots, a_n\}$.
2. $A \to \beta_1 \cdot a\beta_2[a_1, \ldots, a_n]$ ist ein shift Schritt, falls das a das Vorschauzeichen ist.
3. $S \to \alpha \cdot [\$]$ ein reduce Schritt mit $S \to \alpha$ akzeptieren, falls das Vorschauzeichen \$ ist.

Beispiel:

Sei $S \to A$, $A \to BA \mid \epsilon$, $B \to aB \mid b$ eine Grammatik G. Dann:



Hieraus ergibt sich folgender DFA:



Die zugehörige Parsing Tabelle für LR(1) (In x-Achse die Zustände, in y-Achse das Vorschauzeichen)

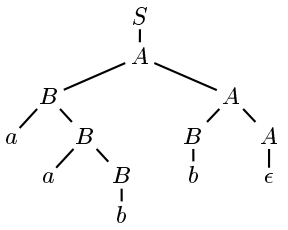
	a	b	\$	S	A	B
0	$S3$	$S4$	$A \to \epsilon$	–	1	2
1	–	–	$S \to A$	–	–	–
2	$S3$	$S4$	$A \to \epsilon$	–	5	2
3	$S3$	$S4$	–	–	–	6
4	$B \to b$	$B \to b$	$B \to b$	–	–	–
5	–	–	$A \to BA$	–	–	–
6	$B \to aB$	$B \to aB$	$B \to aB$	–	–	–

Für $aabb\$$ ergibt sich folgende Ableitung.

Keller	Resteingabe	Zustand	Aktion	
0	$aabb\$$	S	$0a = 3$	
$0a3$	$abb\$$	S	$3a = 3$	
$0a3a3$	$bb\$$	S	$3b = 4$	
$0a3a3\underline{b4}$	$b\$$	R	$B \to b$	$3B = 6$
$0a3\underline{a3B6}$	$b\$$	R	$B \to aB$	$3B = 6$
$0\underline{a3B6}$	$b\$$	R	$B \to aB$	$0B = 2$
$0B2$	$b\$$	S	$2b = 4$	
$0B2\underline{b4}$	\$	R	$B \to b$	$2B = 2$
$0B2B2\underline{\epsilon}$	\$	R	$A \to \epsilon$	$2A = 5$
$0B2\underline{B2A5}$	\$	R	$A \to BA$	$2A = 5$
$0\underline{B2A5}$	\$	R	$A \to BA$	$0A = 1$
$0\underline{A1}$	\$	R	$S \to A$	
$0S$	\$		akzeptiert	

$$S \to A \to BA \to BBA \to BB\epsilon \to Bb \to aBb \to aaBb \to aabb$$

Und es ergibt sich folgender Ableitungsbaum:



Anmerkungen

1. LR(1) Sprache $\iff$ DCFC
2. bei umfangreichen DFA existieren Zustände mit der selben Menge, aber verschiedenen Vorschaumengen. Bilde einen Zustand der item Menge mittels Vereinigung der Vorschaumengen. Falls ein reduce/reduce Konflikt entsteht, bile keine Vereinigung. (look-ahead-LR)
3. Die Software für das Erzeugen von LR(1) Parsing Tabellen heist YACC (yet another compiler compiler)

Kapitel 5

Deterministische Kellerautomaten

Es gilt $\{\text{DCFL}\} \subsetneq \{\text{CFL}\}$. Sei $L_2 := \left\{x\tilde{x} \,\middle|\, x \in \Sigma^+\right\}$ mit $\Sigma = \{0,1\}$ und $\tilde{x}$ die Spiegelung von x, die Sprache der Palindrome [1]. **Behauptung:**

$$L_2 \in \left\{\text{CFL}\right\} - \left\{\text{DCFL}\right\}$$

L_2 ist eine CFL, da $S \to 0S0 \,|\, 1S1 \,|\, 00 \,|\, 11$

Definition 5.1 ((Inverser) Homomorphismus) *1. Seien Σ und Δ disjunkte Alphabete. Die Abbildung h mit $h : \Sigma \to \Delta^*$ mit $h(\epsilon) := \epsilon$ bzw. $h(xa) := h(x)h(a)$ heist* ***Homomorphismus***

Sei $L_0 \subseteq \Sigma^*$, dann heist $h(L_0) := \left\{h(x) \,\middle|\, x \in L_0\right\} \subseteq \Delta^*$ das ***homomorphes Bild****.*

2. Sei das Wort $w \in \Delta^*$ bzw. $S \subseteq \Delta^*$. Die Summe $h^{-1}(w) := \left\{x = \Sigma^* \,\middle|\, h(x) = w\right\}$ heist ***homomorphes Urbild***[2] *von w. $h^{-1}(S) := \left\{x \in \Sigma^* \,\middle|\, h(x) \in S\right\}$ heist homomorphes Urbild von S. Die Abbildung h^{-1} heist* ***inverser Homomorphismus***

Beispiel:

Sei $\Sigma = \{0,1\}$ und $\Delta = \{a,b\}$. $h : 0 \mapsto aa$ und $h : 1 \mapsto aba$

1. $h(010) = h(0)\,h(1)\,h(0) = aaabaaa$
2. $L_1 = (01)^*$, dann $h(L_1) = (aaaba)* = \{\epsilon, aaaba, aaabaaaaba, aaabaaaabaaaaba, \dots\}$
3. Sei $S := \{ab, ba\}^*a \subseteq \Delta^*$, also $S = \{a, aba, baa, ababa, abbaa, baaba, babaa, \dots\}$, dann ist $h^{-1}(S) = \{1\}$.

Denn es gilt:
$h(0) = h(1) = a\cdots$ fängt mit a an, also $h(x) = a\cdots$ für $x \in \Sigma^* - \{\epsilon\}$. $w \in S$ ist unter h nur ein Bild für $w = a\cdots$, somit entweder $w = a$ oder $w \in ab\{ab,\, ba\}^*a$

Fall 1 mit $w = a$ ist nicht möglich, weil $\forall x : |h(x)| \geq 2$. Im Fall 2 gilt $w = abw'$ mit $w' \in \{ab,\, ba\}^*a$, folglich muss $h^{-1}(w) = 1\cdots$ gelten. Wegen $h(1) = aba$ ist aber $w' = a\cdots$. Also gilt wiederum entweder $w' = a$ oder $w' = au$ mit $|u| \geq 1$.

Wenn $w' = a$, dann ist $w = aba \in S$, also $h^{-1}(w) = 1$. Falls $w' = au \in \left\{\{ab,\, ba\}^*a\right\}$ ist $w' = abw''$ und somit $w = ababw''$. Für kein $x \in \Sigma^*$ gilt dann aber $h(x) = w$, da stets $x = 1y$. Also entweder

[1] In unserem Fall nur die Wörter mit 2n Zeichen
[2] Evtl. gilt $h^{-1}(w)\emptyset$

$x = 1\underbrace{0z}_{y}$, dann aber $h(x) = abaaa\cdots$ oder $x = 1\underbrace{1z}_{y}$ bzw. $h(x) = abaaba\cdots$, die beide nicht erzeugt werden können.

Lemma 5.2 *DCFLs sind abgeschlossen gegenüber dem inversen Homomorphismus.*

Beweis
[Struktur noch unklar]

1. Sei $L_0 \subseteq \Delta^*$ eine DCFL. Zu zeigen ist, dass aus $h : \Sigma \to \Delta^*$ Homomorphismus folgt, dass $h^{-1}(L_0) \subseteq \Sigma^*$ eine DCFL ist, d.h. dass ein DPDA M' mit $h^{-1}(L_0) := \{x \in \Sigma^* \mid h(x) \in L_0\}$ und einem Sprachschatz $T(M')$ existiert.

   Nach Voraussetzung ist L_0 eine DCFL, es existiert also ein DPDA M mit $L_0 = T(M)$, mit M gegeben. Es ist also zu zeigen, dass für $x \in \Sigma^*$ ein DPDA M' existiert mit

   $$x \in T(M') \iff h(x) \in T(M)$$

   Konstruktionsidee für den M':
   Gegeben sei ein $x = a_1 a_2 \ldots a_n \in \Sigma^*$, bilde $h(x) = h(a_1)h(a_2)\ldots h(a_n) \in \Delta^*$ und definiere M' so, dass $x \in T(M') \iff h(x) \in M$.

   h ist ein Homomorphismus, betrachte dann die Verarbeitung eines $h(a)$ durch M für ein $a \in \Sigma$.

   Die Verarbeitung eines $x \in \Sigma^*$ durch M' geschieht folgendermaßen:

   (a) M' liest das nächste Zeichen a aus x und erzeugt daraus $h(a) \in \Delta^*$. M' agiert also lesend.

   (b) M' simuliert die Verarbeitung des $h(a) \in \Delta^*$ durch M ohne weiterlesen in x. M' wechselt also spontan seinen Zustand.

   Bei Verarbeiten des $h(a)$ durch M (teils lesend, teils spontan) wird Keller des M benötigt. [Was passiert also mit dem Keller des M'? Wie geschieht die Speicherung des (Restes von) $h(a)$?]
   Der Keller des M' entspricht dem Keller des M. Der $h(a)$-Rest ist im M' Zustand "abgespeichert".

2. M' sieht folgendermaßen aus:

   $$Q' := \{[q, w] \mid q \in Q, w \in \Delta^*, h(a) - \text{Rest}\} \quad F' := \{[p, \epsilon] \mid p \in F\} \quad q_0' := [q_0, \epsilon]$$

   Σ, Γ und z_0 wie bei M und δ' des M'. Dann

   (a) $\delta'([q, w], \epsilon, Y) = ([p, w], \gamma) \iff \delta(q, \epsilon, Y) = (p, \gamma)$
   Dies ist ein spontaner M-Übergang simuliert durch einen spontanen M'-Übergang, mit der selben Wirkung im Keller und einem Verkürzen des Restes. (Y wird durch γ ersetzt)

   (b) $\delta'([q, dw], \epsilon, Y) = ([p, w], \gamma) \iff \delta(q, d, Y) = (p, \gamma)$
   Dies entspricht einem lesenden M-Übergangs simuliert durch einen spontanen M'-Übergang, mit der selben Wirkung im Keller und einem Verkürzen des Restes.

   (c) $\delta'([q, \epsilon], aY) = ([q, h(a)], Y)$

Lesender M'-Übergang ändert nur den M'-Zustand, wobei M'-Keller unverändert bleibt.

M' ist deterministisch, denn ein lesen findet nur im Zustand $[q, \epsilon]$, ein spontaner Übergang nur im $[q, u]$ $(u \neq \epsilon)$ Zustand statt. Die zwei Arten der spontanen Übergänge bei M' sind klar unterscheidbar über den DPDA M.

Für ein $\alpha \in \Sigma$ gilt als:

$$\big(q, h(a), \alpha\big) \overset{*}{\underset{M}{\longmapsto}} \big(t, \epsilon, \beta\big) \quad \Longleftrightarrow \quad \big([q, \epsilon], a, \alpha\big) \overset{2c}{\underset{M}{\longmapsto}} \big([q, h(a)], \epsilon, \alpha\big) \overset{2a\,2c}{\overset{*}{\underset{M}{\longmapsto}}} \big([t, \epsilon], \epsilon, \beta\big)$$

Es bleibt also noch zu zeigen:

$$h(x) \in T(M) \iff x \in T(M')$$

$\Longleftarrow$ $h(x) \in T(M)$, also $\big(q_0, h(x), z_0\big) \overset{*}{\underset{M}{\longmapsto}} (p, \epsilon, \beta)$ mit $p \in F$.

Über Teilabschnitte $\big(q_i, h(a_i), \alpha\big) \overset{*}{\underset{M}{\longmapsto}} (f_i, \epsilon, \beta)$, $\alpha \in \Sigma$ hintereinander, denn $x = a_1 \dots a_n$, $h(x) = h(a_1) \dots h(a_n)$ folgt, dass $h(a_{i+1})$ erst verarbeitet wird, nachdem $h(a_i)$ verarbeitet ist. Der Rest von $h(a_i)$ entspricht dabei Σ.

Das Simulieren jeder Teilabschnittsverarbeitung ergibt die Simulation der gesamten Verarbeitung durch M als $\big([q_0, \epsilon], x, z_0\big) \overset{*}{\underset{M}{\longmapsto}} \big([p, \epsilon], \epsilon, \beta\big)$ mit $[p, \epsilon] \in P$.

$\Longrightarrow$ Aus $x = a_1 \dots a_n \in T(M')$ folgt $(p_n \in F)$:

$$\begin{aligned}
\big([q_0, \epsilon], a_1 \dots a_n, z_0\big) &\overset{2c}{\underset{M'}{\longmapsto}} \big([q_0, h(a_1)], a_2 \dots a_n, z_0\big) \\
&\overset{2a2b*}{\underset{M'}{\longmapsto}} \big([p_1, \epsilon], a_2 \dots a_n, \alpha_1\big) \\
&\overset{2c}{\underset{M'}{\longmapsto}} \big([p_1, h(a_2)], a_3 \dots a_n, \alpha_1\big) \\
&\overset{2a2b*}{\underset{M'}{\longmapsto}} \big([p_1, \epsilon], a_3 \dots a_n, \alpha_2\big) \\
&\longmapsto \dots \\
&\overset{2a2b*}{\underset{M'}{\longmapsto}} \big([p_n, \epsilon], \epsilon, \alpha_n\big)
\end{aligned}$$

Insgesamt ergibt sich also in M $(i \in \{1, \dots, n-1\})$:

$$\big(q_0, h(a_1), z_0\big) \overset{*}{\underset{M'}{\longmapsto}} (p_1, \epsilon, \alpha_1) \quad \text{und} \quad \big(p_i, h(a_{i+1}), \alpha_i\big) \overset{*}{\underset{M'}{\longmapsto}} (p_{i+1}, \epsilon, \alpha_{i+1})$$

Also

$$\big(q_0, h(a_1 \dots a_n), z_0\big) \overset{*}{\underset{M'}{\longmapsto}} \big(p_n, \epsilon, \alpha_n\big)$$

und somit $h(x) \in T(M)$.

■

Lemma 5.3 *DCFLs sind abgeschlossen, gegen den Durchschnitt mit der Menge der regulären Sprachen.*

Beweis
[mit Hilfe von Automaten] Sei $M = \big(Q_M, \Sigma, \Gamma, \delta_M, q_0, z_0, F_M\big)$ ein DPDA und $L_0 = T(M)$ die zugehörige DCFL und $A = \big(Q_A, \Sigma, \delta_A, p_0, F_A\big)$ ein DFA mit zugehöriger regulärer Sprache $R_0 = T(A)$.

Die Beweisidee liegt in einem DPDA M', der ein $w \in \Sigma^*$ parallel durch M und A verarbeitet. Ein Wort w wird dann akzeptiert, wenn sowohl M als auch A das Wort w akzeptieren.

Sei $Q' := Q_A \times Q_M$; $[p, q]$ mit $p \in Q_A$ und $q \in Q_M$. Weiterhin $q_0' = [p_0, q_0]$ und $F' = F_A \times F_M$. Dann

$$\delta'\big([p,q], a, Y\big) = \big([p', q'], \gamma\big) \iff \delta_A(p,a) = p' \wedge \delta_M(q, a, Y) = (q', \gamma)$$

Der Keller des M' entspricht also genau dem Keller des M. M' ist deterministisch, weil M und A beide deterministisch sind. Ohne Beweis gelte:

$$\big([p_0,q_0], w, z_0\big) \overset{i}{\underset{M'}{\vdash}} \big([p,q], \epsilon, \gamma\big) \iff (q_0, w, z_0) \overset{i}{\underset{M}{\vdash}} (q, \epsilon, \gamma) \wedge \delta_A(p_0, w) = p$$

und damit

$$\begin{aligned} w \in T(M') &\iff \big([p_0,q_0], w, z_0\big) \overset{i}{\underset{M'}{\vdash}} \big([p_f, q_f], \epsilon, \gamma\big) \\ &\iff \delta_A(p_0, w) = p_f \text{ und } (q_0, w, z_0) \overset{i}{\underset{M}{\vdash}} (q_f, \epsilon, \gamma) \\ &\iff w \in T(A) \wedge w \in T(M) \\ &\iff w \in L_0 \cap R_0 \end{aligned}$$

■

Definition 5.4 (MIN) *Sei L_0 eine DCFL, dann ist*
$\mathrm{MIN}(L_0) := \big\{x \in L_0 \,\big|\, \textit{Kein echter Anfang } w \textit{ des } x \textit{ ist in } L_0\big\}$

Lemma 5.5 *L_0 DCFL $\Longrightarrow$ $\mathrm{MIN}(L_0)$ DCFL*

Beweis
Die Idee hierbei ist, dass das Akzeptieren von w und $x = wz$ mit $z \neq \epsilon$, durch einen DPDA nur bei Übergängen ab einem Endzustand möglich ist. Streiche also in M alle Übergänge ab dem Endzustand. M' bleibt dann ein DPDA.

Sei $L_0 = T(M)$ mit M DPDA. Wähle M' wie M aber ohne Übergänge $\delta'(q, a, z)$ für $q \in F$. Dann gilt

$$x \in \mathrm{MIN}(L_0) \iff x \in T(M')$$

$\Longrightarrow$ $x \in \mathrm{MIN}(L_0)$, also $x \in L_0$. Dann existiert kein $w \in L_0$ mit $x = wz$, $z \neq \epsilon$, dass $(q_0, x, z_0) \overset{i}{\underset{M}{\vdash}} (q_f, \epsilon, \gamma)$ mit $q_f \in F$, da schon $x \in L_0$.

Zwischen q_0 und q_f liegen keine Endzustände, dasselbe gilt auch für M', also gilt $x \in T(M')$.

$\Longleftarrow x \in T(M') \Longrightarrow (q_0, x, z_0) \overset{i}{\underset{M}{\vdash}} (q_f, \epsilon, \gamma)$

Der selbe Übergang existiert aber auch in M. Es liegen also keine Endzustände zwischen q_0 und q_f. Für ein w mit $x = wz$, $z \neq \epsilon$ gilt also $w \notin T(M)$. Somit $x \in \mathrm{MIN}(L_0)$.

■

Satz 5.6 *Die Sprache* $L_1 = \left\{ x\tilde{x} \,\middle|\, x \in \{0,1\}^+ \right\}$ *ist eine kontextfreie, aber keine deterministisch kontextfreie Sprache.*

Beweis

1. L_1 ist eine kontextfreie Sprache
2. Annahme: L_1 ist eine deterministisch kontextfreie Sprache
   (a) $R_0 := (01)^* (10)^* (01)^* (10)^*$ ist eine reguläre Sprache. Sei $L_2 := L_1 \cap R_0$ und es gilt:
   Wenn L_1 DCFL, dann auch L_2 DCFL.
   (b) Dann ist $L_2 = \left\{ (01)^i (10)^j (01)^j (10)^i \,\middle|\, i, j \geq 0, i+j > 0 \right\}$. Sei $L_3 = \mathrm{MIN}(L_2)$ wobei gilt:
   Wenn L_2 DCFL, dann folgt L_3 DCFL.
   (c) $L_3 = \left\{ (01)^i (10)^j (01)^j (10)^i \,\middle|\, 0 \leq j < i \right\}$
      i. Für $j \geq i$ existiert ein $w \in L_2$ mit $x = wz$, $z \neq \epsilon$. Aus $j \geq i$ folgt $j = i + \nu$ mit $\nu \geq 0$, dann aber $x = \underbrace{(01)^i (10)^i}_{w \in L_2} \underbrace{(10)^\nu (01)^j (10)^i}_{z \neq \epsilon}$ im Widerspruch zur Annahme.
      ii. Für $j < i$ existiert kein Präfix des $x \in L_2$. Aus $j < i$ folgt $i = j + \mu$ mit $\mu \geq 1$ und somit $x = (01)^j (01)^\mu (10)^j (01)^j (10)^j (10)^\mu$. Also kein echter Anfang des x ist ein Palindrom
   (d) Sei $h : \{a, b\} \to \{0,1\}^*$ ein Homomorphismus mit $h(a) = 01$ und $h(b) = 10$, dann $L_4 := h^{-1}(L_3)$ und falls L_3 eine DCFL ist auch L_4 eine DCFL.
   (e) Dann gilt $L_4 = \{a^i b^j a^j b^i \mid 0 \leq j < i\}$. L_4 ist aber nicht kontextfrei (siehe Übung), also auch keine DCFL

L_1 ist keine DCFL. ■

Kapitel 6

Transition Networks

Syntaktische Analyse einer CFL mit endlichen Automaten, welcher *Selbsteinbettung* zulässt.

6.1 Selbsteinbettung

Sei $\{a^n b^n \mid n \geq 1\}$ bzw. $S \to aSb \mid ab$, dann gilt $S \overset{+}{\Longrightarrow} a^n S b^n$.

Definition 6.1 *Sei $G = (V, T, P, S)$ eine kontextfreie Grammatik, dann gilt:*

1. *$A \in V$ heist **selbsteinbettend**, gdw. $A \overset{*}{\Rightarrow} \alpha A \beta$ mit $\alpha \neq \epsilon$, $\beta \neq \epsilon$.*
2. *G heist **selbsteinbettend**, gdw. in V selbsteinbettende non terminals existieren.*
3. *Eine kontextfreie Sprache L_0 heist **selbsteinbettend**, gdw. jedes G mit $L_0 = L(G)$ selbsteinbettend ist.*

Satz 6.2 *Sei L_0 eine kontextfreie Sprache, dann gilt:*

$$L_0 \text{ regulär} \iff L_0 \textbf{ nicht } \text{selbsteinbettend}$$

Definition 6.3 *Eine Produktion P heist **zusammenhängend** (zh), gdw. $\forall A, B \in V$ ist $A \overset{*}{\Rightarrow} \alpha B \beta$ (α, β evtl. ϵ)*

Beweis

$\Longrightarrow$ L_0 ist regulär, also existiert eine Typ 3 Grammatik für L_0. Diese ist aber nie selbsteinbettend.

$\Longleftarrow$ L_0 ist nicht selbsteinbettend. Dann existieren Produktion, welche L_0 erzeugen aber nicht selbsteinbettend sind. Also ist kein $A \in V$ selbsteinbettend.

Es gilt:

$$P \text{ ist nicht selbsteinbettend} \iff \big(P \text{ nicht selbsteinbettend } \wedge P \text{ zh }\big) \vee \quad (6.1)$$
$$\big(P \text{ nicht selbsteinbettend } \wedge P \text{ nicht zh }\big) \quad (6.2)$$

zu (6.1) Zeige: falls P nicht selbsteinbettend aber zusammenhängend ist folgt, dass P eine Typ 3 Sprache ist.

i. Zeige, dass P nur Produktionen der Form $A \to \alpha B$ oder $A \to B\alpha$ besitzt.

Annahme: $A \to \alpha B\beta$ mit $\alpha \neq \epsilon \neq \beta$

Dann gilt: $B \overset{*}{\Rightarrow} \gamma A\delta$, also $A \overset{*}{\Rightarrow} \underbrace{\alpha\gamma}_{\neq\epsilon} A \underbrace{\delta\beta}_{\neq\epsilon}$.

A ist also selbsteinbettend im Widerspruch zur Voraussetzung.

ii. Zeige, dass P keine Produktion der Form $A \to \alpha B$ und [gleichzeitig] $C \to D\beta$ mit $\alpha \neq \epsilon \neq \beta$ besitzt.

Dann gilt: $A \Longrightarrow \alpha B \overset{*}{\underset{\text{zh}}{\Rightarrow}} \alpha\gamma C\delta \Longrightarrow \alpha\gamma D\beta\delta \overset{*}{\underset{\text{zh}}{\Rightarrow}} \underbrace{\alpha\gamma\kappa}_{\neq\epsilon} A \underbrace{\lambda\beta\delta}_{\neq\epsilon}$.

Folglich wäre A selbsteinbettend im Widerspruch zur Voraussetzung.

Aus i. und ii. folgt, dass P vom Typ 3 ist.

zu (6.2) Induktion über $|V|$

Induktionsanfang

$|V| = 1, V = \{S\}$, da nicht zusammenhängend existiert nur $S \to x$ mit $x \in T^*$. Die Sprache L_0 ist also endlich, folglich eine Typ-3 Sprache.

Induktionsvorraussetzung

P [produziert?] ist eine Typ-3 Sprache für $|V| \leq n - 1$

Induktionsschluß

P ist nicht zusammenhängend, dann gilt speziell für ein $A \in V$:
nicht $A \overset{*}{\Rightarrow} \alpha S\beta$, denn sonst[1] würde stets gelten:

$$A \overset{*}{\Rightarrow} \alpha S\beta \Longrightarrow \alpha\gamma B\delta\beta$$

im Widerspruch zu nicht zusammenhängend.

Nun müssen noch 2 Fälle unterschieden werden:

$A \neq S$ Streiche $A \to \gamma$ aus P. Sei $T' := T \cup \{a_0\}$ mit $a_0 \notin T$. Ersetzte A rechts durch a_0, dann $P \to P'$ mit $|V'| = n - 1$ und P' nicht zusammenhängend.

Nun gilt **entweder** P' zusammenhängend, dann tritt aber 6.1 ein und die Sprache L_0' ist eine Typ-3 Sprache, **oder** P' ist nicht zusammenhängend, dann ist aber nach Induktionsvorraussetzung L_0' vom Typ-3.

$A = S$ Betrachte $K := \left\{y \,|\, A \overset{*}{\underset{P}{\Rightarrow}} y\right\}$: K wird durch P erzeugt, außer über $S \to \gamma$, d.h. für $K \leq n-1$ NTS ???, d.h. K ist ein Typ-3 Sprache. L_0 entsteht aus L_0' durch Substitution von K für a_0. Folglich ist (L_0' Typ-3, K Typ-3) L_0 Typ-3

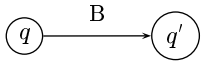
Es gelte $S \overset{*}{\not\Rightarrow} \alpha S\beta$ mit $S \to \beta_1 \mid \ldots \mid \beta_r$.
Sei $K_i := \{x | \beta_i \overset{*}{\Rightarrow} x\}$ mit $\beta_i = B_1^{(i)} \ldots B_1^{(i)}$ und $L_j := \{w | B_j^{(i)} \overset{*}{\Rightarrow} w\}$.
$L_j^{(i)}$ erzeugt durch P, außer $S \to \beta_k$ mit $k \in \{1, \ldots, r\}$ (sonst $S \overset{*}{\Rightarrow} \alpha S\beta$).
Folglich ist $L_j^{(i)}$ eine Typ-3 Sprache. $L_0 = \bigcup_{i=1}^{r} K_i$ mit K_I Konkatenation der $L_j^{(i)}$, also L_0 regulär.

■

[1] Jedes $B \in V$ erreichbar von S aus

6.2 Basic Transition Networks (BTN)

Seien A und B Automaten. Die Selbsteinbettung von B in A geschieht dann folgendermaßen:



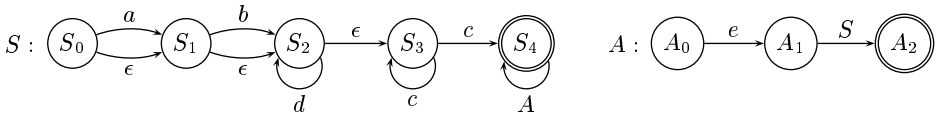
Basic Transition Networks sind äquivalent zu kontext freien Grammatiken.

Beispiel:

BTN $\mapsto$ CFG
Sei $L(S) = \{\epsilon, a, b, ab\} \cdot \{d\}^* \cdot \{c\}^+ \cdot \{L(A)\}^*$ und $L(A) = \{e\} \cdot L(S)$
Damit ergibt sich folgende Grammatik

$$S \to BDCA' \qquad B \to a \,|\, b \,|\, ab \,|\, \epsilon \qquad D \to dD \,|\, \epsilon$$
$$C \to cC \,|\, c \qquad A' \to AA' \,|\, \epsilon \qquad A \to eS$$

und folgender Automat



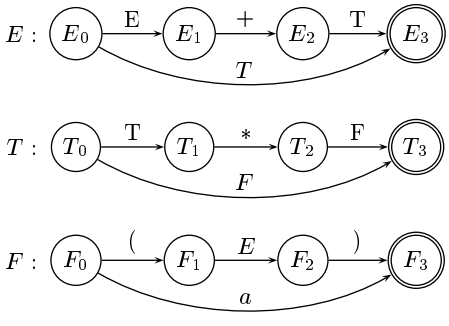
Ableitung:

$Wort$	$Zustand$		
$adcec$	S_0	c	A_1
$dcec$	S_1	c	S_0
$dcec$	S_2	c	S_1
cec	S_2	c	S_2
cec	S_3	c	S_3
ec	S_4	ϵ	S_4
ec	A_0	ϵ	A_2
		ϵ	S_4

(6.3)

Beispiel:

CFG mit $E \to E + T \,|\, T \quad T \to T * F \,|\, F \quad F \to (E) \,|\, a$



$a + a * a$	E_0	$a * a$	T_0
$a + a * a$	E_0	$a * a$	F_0
$a + a * a$	T_0	$*a$	F_3
$a + a * a$	F_0	$*a$	T_3
$+a * a$	F_3	$*a$	T_1
$+a * a$	T_3	a	T_2
$+a * a$	E_3	a	F_0
$+a * a$	E_1	ϵ	F_3
$a * a$	E_2	ϵ	T_3
$a * a$	T_0	ϵ	E_3

Kapitel 7

Das λ Kalkül

7.1 Vorbemerkungen

Sei $f : A \to B$ eine Funktion.

A	B
a_1	b_1
a_2	b_2

heist dann der **Graph** der Funktion.

Wenn $f_1(x) = x + 1$ oder $f_2(x, y) = x + y$ ist die Zuordnung von x und y (z.B. $f_2(2, 3)$) klar. Was ist aber mit $(x + y)(2, 3)$ oder $(x + y)(2)$?

Man schreibt z.B. $(\lambda x.x + y)2 = 2 + y$ oder $(\lambda x.\lambda y.x + y)23 = 2 + 3$.

Weiterhin entspricht $f : A \to B \to C$ dem Ausdruck $f : A \to (B \to C)$. Dies ist auch unter dem Begriff **Currysierung** nach H.B. Curry bekannt.

Desweiteren: $f : N \times N \to N$ mit $x \in N$ folgt z.B. $f_x(y) = x + y$.
Aus $f : A \times B \times C \to D$ folgt $f : A \to (B \to (C \to D)))$.

Schreibkonvention

1. $\to$ assoziiert nacht rechts, es gilt also $A \times B \times C \to D \equiv f : A \to (B \to (C \to D)))$
2. Keine Klammern bei Funktion und Argumenten, also $f\,a\,b := f(a, b)$

7.2 Formales λ Kalkül

Definition 7.1 *Seien e, e_1 und e_2 λ **Ausdrücke**. Die Syntax eines λ Ausdrucks e ist dann wie folgt geben:*

$$e \;\to\; 1 \textit{ Bezeichner} \mid \underbrace{e_1 e_2}_{\textit{Applikation}} \mid \lambda x.e_1 \mid (e_1)$$

Weitere Schreibkonventionen:

1. Der Punkt assoziiert nach rechts, also $\lambda x.\lambda y.e := \lambda x.(\lambda y.e)$.
2. Die Applikation assoziiert nach links, also $(\lambda x.e)(e_1)(e_2) := \big((\lambda x.e)(e_1)\big)(e_2)$.
3. [Die Applikation assoziiert stärker als der Punkt, also $\lambda x.e_1e_2 := \lambda x.(e_1e_2)$. ?]

Definition 7.2 *Der **Gültigkeitsbereich** (scope) von $\lambda.e$*

- *ist der Ausdruck e, ausgenommen Teile von e der Form $(\lambda x.e')$ [sofern das selbe x gemeint ist]!*
- *oder geht, wenn e hinter dem Punkt beginnt, bis zur entsprechenden schließenden Klammer ")", falls vor λ ein öffnende Klammer "(" steht (oder stehen würde), bzw. bis zum Ende.*

Definition 7.3 *Ein x heist **gebunden** in $(\lambda x.e)$, falls x im Gültigkeitsvereich des $\lambda x.$ steht. Ansonsten ist x **frei**.*

Beispiel:

---

1. $\lambda x.ax)x$. x ist frei in ax, aber gebunden in $(\lambda x.ax)$. In a und x "alleine" ist x frei.
2. $(\lambda p.\boxed{(\lambda q.\boxed{(\lambda p.\boxed{q(qp)})})(\lambda r.\boxed{+pr})(+p4)})2$

Definition 7.4 *Seien e, e' Ausdrücke. Dann wird $(\lambda x.e)e'$ gelesen als:* ersetze jedes freie x in e durch e'. *Alternativ wird auch die Operation **Substitution** $[e'/x]e$ verwendet.*

Anmerkung

1. Innerhalb eines "inneren" $(\lambda x.e_1)$ in e ist x **nicht** frei.
2. Beim Ersetzen verschwindet λx

Definition 7.5 *Für die **Substitution** gilt:*

1. $[e'/x]e := e'$ *für* $e = x$ *bzw.* $(\lambda x.x)e' = e'$.
2. $[e'/x]e := e$ *falls x in e nicht frei vorkommt.*
3. $[e'/x]e_1[e'/x]e_2$ *steht für* $e = e_1e_2$.
4. $\big(\lambda x.\underbrace{(\lambda x.e_1)}_{e}\big)\,e' = \lambda x.e_1$
5. (a) $\big(\lambda x.\underbrace{(\lambda y.e_1)}_{e}\big)e' = \lambda y.[e'/x]e_1$ *falls y **nicht** frei in e' ist.*

   (b) *Falls y frei in e' ist, ersetze y in $\lambda y.e_1$ durch einen anderen Bezeichner z und ersetze danach alle x durch e'.*

Beispiel:

1. $(\lambda x.\underbrace{(+x)3}_{e})5 = (+5)3$
2. $(\lambda g.\underbrace{g5}_{e})\underbrace{(\lambda x.(+x)3)}_{e'} = (\lambda x.(+x)3)5 = (+5)3$
3. $(\lambda x.+x)\,3\,5 = +\,3\,5$
4. $$\begin{aligned}(\lambda p.(\lambda q.\underbrace{(\lambda p.q(qp))}_{e})\underbrace{(\lambda r.+p\,r)}_{e'}(+p\,4))\,2 &= (\lambda p.(\lambda q.\underbrace{(\lambda s.q(qs))}_{e})\underbrace{(\lambda r.+p\,r)}_{e'}(+p\,4))\,2\\ &= (\lambda p.(\lambda s.\underbrace{(\lambda r.+p\,r)((\lambda r.+p\,r)s)}_{e})\underbrace{(+p\,4)}_{e'})\,2\\ &= (\lambda p.(\lambda r.\underbrace{+p\,r}_{e})\underbrace{(+p\,(+p\,4))}_{e'})\,2\\ &= (\lambda p.+p\,(+p\,(+p\,4)))2\\ &= +2(+2(+\,2\,4))\end{aligned}$$

Konversionsregeln für λ Ausdrücke

α $\lambda x.e \xrightarrow{\alpha} \lambda y.[y/x]e$ falls y nicht frei in e. Ansonsten konsistente Umbenennung.

β $\underbrace{(\lambda x.e)e'}_{\beta\text{ Redex}} \xrightarrow{\beta} [e'/x]e$ "β-Reduktion", welche aber auch in Gegenrichtung funktionieren.

μ $\underbrace{\lambda x.ex}_{\mu\text{ Redex}} \xrightarrow{\mu} e$, falls x nicht frei in e.
Dann $(\lambda x.ex)y = ey$ wegen $(\lambda x.ex)y = [y/x]ex = [y/x]e[y/x]x = ey$

7.3 Normalformen und Konfluenz

Definition 7.6 *Ein λ-Ausdruck ist ein* ***Normalform****, gdw. e keine β- / μ- Redexe besitzt.*

Beispiel:

β - Reduktion: $(\lambda x.\lambda y.y)ab = \big((\lambda x.(\lambda y.y))a\big)b \xrightarrow{\beta} (\lambda y.y)b \xrightarrow{\beta} b$

μ - Reduktion: $\lambda x.\lambda y.axy = \lambda x.(\lambda yaxy) \xrightarrow{\mu} \lambda x.ax \xrightarrow{\mu} a$

Es gibt nicht immer eine Normalform: $(\lambda x.xx)(\lambda x.xx) \xrightarrow{\beta} (\lambda x.xx)(\lambda x.xx)$

Abhängigkeit von der Reihenfolge:

$$(\lambda x.\underbrace{\lambda y.y}_{e_1})\underbrace{\big((\lambda x.xx)\underbrace{(\lambda x.xx)}_{e_2'}\big)}_{e_1'} b \xrightarrow{\beta} \begin{cases} e_1 b = (\lambda y.y)b \xrightarrow{\beta} b & \text{mit } e_1/e_1' \\ \text{keine Reduktion} & \text{mit } e_2/e_2' \end{cases}$$

7.4 Kombinatoren

Definition 7.7 *Ein* ***Kombinator*** *ist eine λ Ausdruck ohne freie Variablen. Manchmal auch mit* ***geschlossener λ Ausdruck*** *bezeichnet. . Es existieren 6 benannte Kombinatoren:*

Identität: $I := \lambda x.x$

$Ie \to e$, *da* $Ie = (\lambda x.x)e = e$

Projektion $K := \lambda x.\lambda y.x$

$Ke_1e_2 \to e_1$, *da* $Ke = (\lambda x.\lambda y.x)e_1e_2 = (\lambda y.e_1)e_2 = e_1$

Kompositor: $B := \lambda x.\lambda y.\lambda z.x(yz)$

$Be_1e_2e_3 \to e_1(e_2e_3)$, *da*

$Be_1e_2e_3 = \big(\lambda x.\lambda y.\lambda z.x(yz)\big)e_1e_2e_3 = \big(\lambda y.\lambda z.e_1(yz)\big)e_2e_3 = \big(\lambda zee_1(e_2z)\big)e_3 = e_1(e_2e_3)$

Permutierer: $\lambda x.\lambda y.\lambda z.xzy$

$Ce_1e_2e_3 \to e_1e_3e_2$

Verdoppler: $W := \lambda x.\lambda y.xyy$

$We_1e_2 \to e_1e_2e_2$

S Kombinator: $S\lambda.x\lambda y.\lambda z.xz(yz)$

$Se_1e_2e_3 \to e_1e_3(e_2e_3)$

Satz 7.8

1. $I = SKK$
2. $B = S(KS)K$
3. $C = S(BBS)(KK)$

Beweis

1. $SKKl = Kl(KK) = l = Il$
2. $S(KS)Kl_1l_2l_3 = (KS)l_1(Kl_1)l_2l_3 = KSl_1(Kl_1)l_2l_3 =$
   $S(Kl_1)l_2l_3 = (Kl_1)l_3(l_2l_3) = l_1(l_2l_3) = Bl_1l_2l_3$
3. analog

■

Beispiel: Bool'sche Werte, Bedingungen

Sei $\text{true} := \lambda x.\lambda y.y$, $\text{false} := \lambda x.\lambda y.xy$ und $\text{cond} := \lambda x.\lambda y.\lambda z.z(\lambda v.y)x$. Dann gilt:

$$\text{cond}\, e_1 e_2\, \text{true} = e_1 \quad \text{und} \quad \text{cond}\, e_1 e_2\, \text{false} = e_2$$

Denn:
Seien y, z nicht frei in e_1 und v, z nicht frei in e_2.

$$\begin{aligned}\text{cond}\, e_1 e_2 &= \big(\lambda x.\lambda y.\lambda z.z(\lambda v.y)x\big)e_1e_2\\ &= \big(\lambda y.\lambda z.z(\lambda v.y)e_1\big)e_2\\ &= \lambda z.z(\lambda v.e_2)e_1\end{aligned}$$

und somit

$$\begin{aligned}(\text{cond}\, e_1 e_2)\, \text{true} &= \big(\lambda z.z(\lambda v.e_2)e_1\big)(\lambda x.\lambda y.y)\\ &= (\lambda x.\lambda y.y)(\lambda v.e_2)e_1\\ &= (\lambda y.y)e_1 = e_1\end{aligned}$$

bzw.

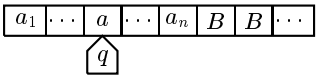
$$\begin{aligned}(\text{cond}\, e_1 e_2)\, \text{false} &= \big(\lambda z.z(\lambda v.e_2)e_1\big)(\lambda x.\lambda y.xy)\\ &= (\lambda x.\lambda y.xy)(\lambda v.e_2)e_1\\ &= (\lambda y.(\lambda v.e_2)y)e_1\\ &= (\lambda v.e_2)e_1 = e_2\end{aligned}$$

Kapitel 8

Turingmaschienen und Unentscheidbarkeit

Wiederholung Turingmaschienen

Eine Standard Turingmaschiene (TM) besteht aus einem einseitig unendlichen Band und einem Lesekopf. Es gilt $(q, a) \mapsto (q', a', m)$ mit $m[ove] \in \{R, L\}$.



Es sind Verlängerungen $(q, a) \mapsto (q', a', m)$ und Verkürzungen $(q, a_n) \mapsto (q', B, m)$ erlaubt. Alternativ kann auch $qaq'a'm$ geschrieben werden.

Eine Konfiguration wird als $a_1 \dots qa \dots a_n$ dargestellt.

Ein Wort wird mit Endzustand $q_f \in F$ akzeptiert, falls $q_0 a_1 \dots a_n \overset{*}{\underset{M}{\longmapsto}} b_1 \dots q_f b \dots b_m$ gilt.

Ein Wort wird nicht akzeptiert, falls der Kopf in einem Zustand $q \notin F$ stehen bleibt oder falls er gar nicht stehen bleibt.

Wiederholung Entscheidbarkeit

$L_0 \subseteq \Sigma^*$ heist genau dann semi-entscheidbar, wenn eine TM M existiert mit $M : L_0 = T(M)$. D.H. bei Eingeben von $x \in L_0$ hält M und ist in einem Endzustand.

$L_0 \in {}^1\Sigma^*$ ist genau dann entscheidbar, wenn ein TM M existiert und M für jedes $x \in \Sigma^*$ anhält und je nach Zustand entscheidet.

L_0 ist also entscheidbar, wenn L_0 und $\bar{L_0}$ beide semi entscheidbar sind.

Beispiel:

$\hat{L} := \{x_i \mid x_i \notin T(M_i)\}$ ist semi entscheidbar. Das Kompliment $\bar{\hat{L}}$ ist nicht semi entscheidbar.

---

[1] $\in$ oder $\subseteq$

Rückführung von Problem P_2 auf P_1

P_1 sei bekannt als unentscheidbar. Dann ist zu zeigen, dass auch P_2 unentscheidbar ist. Ein Beweisgang, kann dann folgendermaßen aussehen:
Es ist zu einem beliebigen P_1 zu zeigen, dass ein spezielles P_2 existiert, so dass gilt: P_1 ist genau dann entscheidbar, wenn auch P_2 entscheidbar ist.
Danach zeigt man, dass P_2 stets entscheidbar ist, also auch ein spezielles P_2 und somit P_1 stets entscheidbar.

8.1 Das Post'sche Korrespondenzproblem (PKP)

Definition 8.1 *Sei ein Alphabet Σ gegeben, dazu zwei gleich lange Stringfolgen $A := (x_1, \ldots, x_k)$ und $B := (y_1, \ldots, y_k)$ mit $x_i, y_j \in \Sigma^+$ (evtl. in A, B Strings mehrfach).*

Es stellt sich die Frage, ob eine (nicht leere) Folge (die "Lösung") $i_1, i_2, \ldots, i_m$ mit $i_\nu \in \{1, \ldots, k\}$ so existiert, dass $x_{i_1} x_{i_2} \ldots x_{i_m} = y_{i_1} y_{i_2} \ldots y_{i_m}$ gilt.

*Das **PKP** besteht dann daraus, ob obige Frage für beliebige Σ, A, B (mit ja oder nein) beantwortbar ist.*

Beispiel:

$\Sigma = \{0, 1\}$, $A = (101, 11, 011)$, $B = (10, 111, 01)$. Eine Lösung wäre z.B. (1,2), denn $\underbrace{12}_{y_1}\underbrace{1\overbrace{11}^{x_2}}_{y_2}$

Sei $A = (10, 010, 0011)$ und $B = (1, 01, 110)$, dann existiert keine Lösung, da $|x_i| > |y_i|$

Sei $A = (10, 011, 101)$ und $B = (101, 11, 011)$.
Dann $i_1 = 1$, da i_1 und i_3 unterschiedlich anfange. Also $A = 10\ldots$ und $B = 101\ldots$.
x_{i_2} muss also mit 1 Anfangen, folglich $i_2 \in \{1, 3\}$
Dann aber entweder $i_2 = 1$ und $x_1 x_1 = 1010 \neq 1011 = y_1 y_1$ oder $i_2 = 3$ und $A = 10101$ bzw. $B = 101011$. Der zweite String ist also stets länger, das Problem also unlösbar.

Satz 8.2 *Das PKP mit $\Sigma = \{a\}$ ist entscheidbar.*

Beweis
Über Fallunterscheidungen ∎

Definition 8.3 (modifiziertes PKP (MPKP)) *Gegeben Σ, $A = x_1 \ldots x_k$, $B = y_1 \ldots y_k$ und dazu vier Strings $a, b, c, d \in \Sigma^*$.*

Ist dann die Frage, ob für beliebige Σ, A, B, a, b, c, d gilt $a x_{i_1} \ldots x_{i_m} c = b y_{i_1} \ldots y_{i_m} d$, entscheidbar.

Satz 8.4 *Das MPKP ist unentscheidbar*

Beweis
Die Beweisidee besteht aus der Rückführung auf das Wortproblem der Typ-0 Sprachen. D.h. starte bei einer *beliebigen* Typ-0 Grammatik G und einem beliebigen String $w \in T^*$ und konstruiere dazu ein *spezielles* MKPK so, dass dieses MPKP genau dann lösbar ist, wenn das Wortproblem (G, w) lösbar ist.

Für ein spezielles MPKP sei $w \in T^*$ gegeben.
Weiterhin $G = (V, T, P, S)$, $P = \{x_i \to y_i | i = 1 \dots p\}$ und $\widetilde{V} = V \cup T \cup \{\$\} = \{\sigma_1, \sigma_2, \dots, \sigma_r\}$.
Sei $A := (x_1, \dots, x_p, \sigma_1, \dots, \sigma_r)$ und $B := (y_1, \dots, y_p, \sigma_1, \dots, \sigma_r)$ mit $a = \epsilon$, $b := \$S$ und $c = \$w$
Dieses MPKP ist genau dann lösbar, wenn $x_{i_1} \dots x_{i_m} \$w = \$S y_{i_1} \dots y_{i_m}$

Hilfsaussage: $S \underset{P}{\overset{*}{\Longrightarrow}} \alpha \iff x_{i_1} \dots x_{i_m} \$\alpha = \$S y_{i_1} \dots y_{i_m}$

Und damit die Rückführung auf $\alpha = w$ ∎

Beweis
[der Hilfsaussage]

1. $S \overset{*}{\Longrightarrow} \alpha$: $S = \alpha_0 \Longrightarrow \alpha_1 \Longrightarrow \dots \alpha_{r-1} \Longrightarrow \alpha_r = \alpha$
   $\varphi := \$\alpha_0 \$ \alpha_1 \$ \dots \$ \alpha_{r-1}$ $\psi := \$\alpha_1 \$ \dots \$ \alpha_{r-1} \$ \alpha_r$
   $\varphi \$ \alpha = \$ S \psi$, also $\varphi = x_{i_1} \dots x_{i_m}$ und $\psi = y_{i_1} \dots y_{i_m}$, wobei $x_\nu \hat{=} y_\nu$ stets in A, B
2. $z := x_{i_1} \dots x_{i_m} \$\alpha = \$ S y_{i_1} \dots y_{i_m}$, also $x_{i_1} = \$ \dots$. Da \$ aber nicht in den Produktionen ist, gilt $x_{i_1} = \$$ und somit $y_{i_1} = \$$

   Dann ist $z = \$ x_{i_2} \dots x_{i_m} \$\alpha = \$S\$ y_{i_2} \dots y_{i_m}$ und analog zu oben $x_{i_2} = S$, also $y_{i_2} = S$ oder $y_{i_2} = \alpha_2$. $S \to \alpha_1$

∎

Index

λ Ausdruck, 53
    geschlossen, 56
λ Kalkül, 53

Analyse
    Bottom-up, 26
    lexikalische, 23
    syntaktische, 24
    Top-Down, 25
Analyseproblem, 24
Applikation, 53
Automat
    äquivalente, 6
    2-Wege, 2
    endlicher, 1
    Mealy, 6
    Moore, 5

Bezeichner, 53
    frei, 54
    gebunden, 54
brauchbarer Anfang, 34
brauchbarere String, 34
BTN, 51

CFG, 23
crossing sequence, 2
    Verträglichkeit, 3

DCFL, 38
DFA, 23

Entscheidbarkeit, 59
Entscheidungszeichen, 31

Gültigkeitsbereich, 54
Grammatik
    LR(0), 37
    LR(1), 39
    sackgassenfrei, 26

Homomorphismus, 43

Identität, 56

Kellerautomat, 24
Kellerautomaten
    deterministische, 43
Kombinatoren, 56
Kompositor, 56
Konfluenz, 56

Lexem, 23
look-ahead, 41
LR(1)-item, 38

Mehrdeutigkeit
    lokale, 27
MIN, 46
MPKP, 60

NFA, 35
Normalformen, 56

Parser, 24
Parsing Tabelle, 38
Permutierer, 56
Phrase, 29
PKP, 60
Postsche Korrespondenzproblem, 60
Präzedenzmatrix, 30
Primphrase, 29
Produktion
    zusammenhängend, 49
Projektion, 56

Randzeichen, 28
reduce, 34

S Kombinator, 56
Schaltnetz, 9
Schaltwerk, 10

Schleifengrammatik, 19
scope, 54
Selbsteinbettung, 49
shift, 34
Sprachen
    LL1, 31
    LR, 34
    Operator, 29
    Präzedenz, 30
Substitution, 54

Token, 23
Transistion Networks, 49
Turingmaschienen, 59

Verdoppler, 56
viable prefix, 34

Wortproblem, 24

YACC, 41

zh, 49