Kellerautomaten und kontextfreie Sprachen

Endliche Automaten reichen zur Beschreibung kontextfreier Sprachen nicht aus.

Bsp. $L = \{w \mid w \in \{a,b\}^* und\ w\ enthält\ ebensoviele\ a\ wie\ b\}$ $L = bbababbabbaaaa$

Nichtdeterministischer Kellerautomat (NKA) (kontextfreie Sprachen)

$M = (Q, \sum, \Gamma, \delta, q_0, k_0, E)$

$Q =$ endliche Menge der Zustände
$\sum =$ Eingabealphabet
$\Gamma =$ Kelleralphabet
$\delta =$ partielle Überführungsfunktion
$q_0 =$ Anfangszustand, $q_0 \in Q$
$k_0 =$ Kellervorbelegungszeichen, $k_0 \in T$
$E =$ Menge von Endzuständen, $E \subseteq Q$

Der NKA $M_1 = (\{q_0, q_1\}, \{a; b\}, \{0, a, b\}, \delta, q_0, 0\ \{q_1\})$ mit folgendem δ

δ (Zustand, Eingabezeichen, oberstes Kellerzeichen) → (Zustand, Eingabezeichen im Keller)

$\delta(q_0, a, b) \ni (q_0, \varepsilon)$

$\delta(q_0, a, 0) \ni (q_0, a0)$

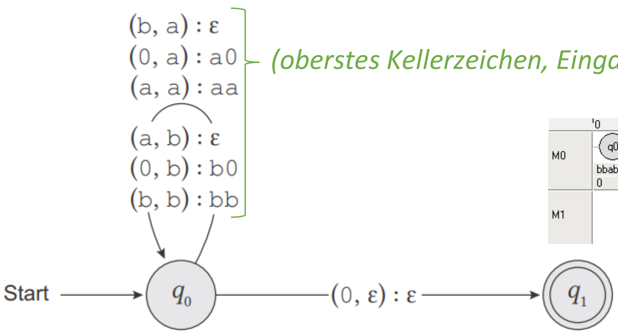
$\delta(q_0, a, a) \ni (q_0, aa)$

$\delta(q_0, b, a) \ni (q_0, \varepsilon)$

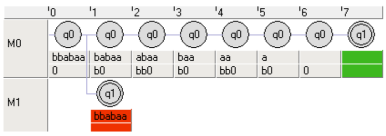
$\delta(q_0, b, 0) \ni (q_0, b0)$

$\delta(q_0, b, b) \ni (q_0, bb)$

$\delta(q_0, \varepsilon, 0) \ni (q_1, \varepsilon)$



(oberstes Kellerzeichen, Eingabezeichen): Kellerwort



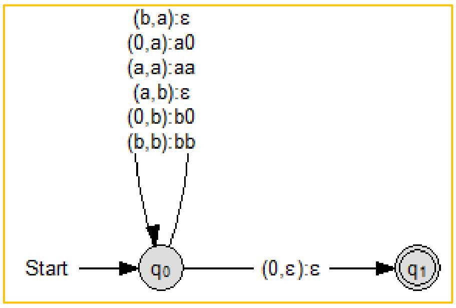
Zustand	Eingabezeichen	Keller (links: top of stack)		
q_0	bbabaa	0	Kellervorbelegungszeichen 0 **(Oberstes Kellerzeichen, Eingabezeichen): Kellerwort**	
q_0	babaa	b0	(0, b): b0	→ (q_0)
q_0	abaa	bb0	(b ,b): bb	→ (q_0)
q_0	baa	b0	(b, a): ε	→ (q_0)
q_0	aa	bb0	(b, b): bb	→ (q_0)
q_0	a	b0	(b, a): ε	→ (q_0)
q_0	ε	0	(b, a): ε	→ (q_0)
q_1	ε	ε	(0, ε): ε	→ (q_1)

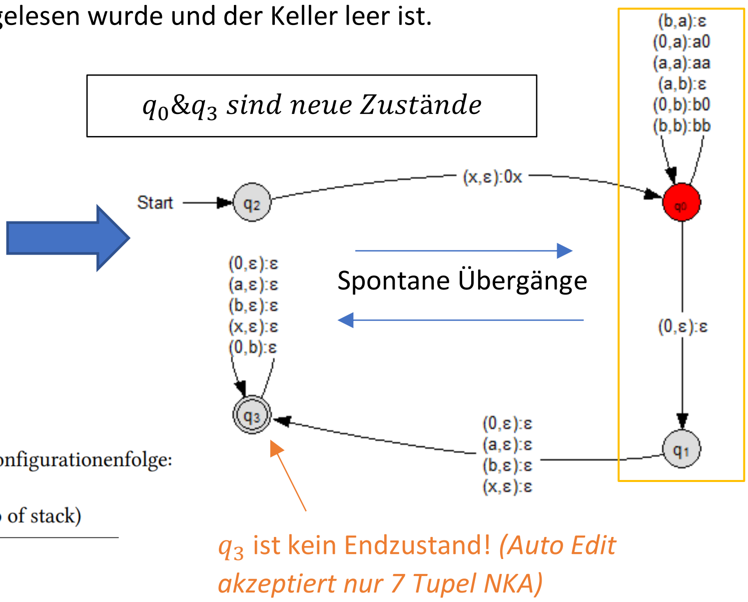
Ein NKA akzeptiert das Eingabewort w genau dann, wenn

a) w vollständig gelesen (abgetastet) wurde und
b) Danach ein Endzustand eingenommen wurde oder nach spontanen Übergängen eingenommen werden kann. Der Kellerinhalt spielt dabei keine Rolle.

NKA 7 Tupel in äquivalenten NKA 6 Tupel Transformieren

Zu jedem 7-Tupel-NKA $M' = (\mathrm{Q}', \sum', \Gamma', \delta', \mathrm{q_0}', \mathrm{k_0}', \mathrm{E}')$ gibt es einen äquivalenten 6-Tupel-NKA $M = (\mathrm{Q}, \sum, \Gamma, \delta, \mathrm{q_0}, \mathrm{k_0})$ (ohne Endzustand E) und umgekehrt. Der Automat akzeptiert ein Eingabewort, wenn es vollständig gelesen wurde und der Keller leer ist.





Für das Wort `bbabaa` ergibt sich dann folgende Konfigurationenfolge:

Zustand	Eingabezeichen	Keller (links: top of stack)
q_2	bbabaa	x
q_0	bbabaa	0x
q_0	babaa	b0x
q_0	abaa	bb0x
q_0	baa	b0x
q_0	aa	bb0x
q_0	a	b0x
q_0	ε	0x
q_1	ε	x
q_3	ε	ε

Äquivalenz von NKA und kontextfreier Grammatik $\mathscr{L}_{\mathrm{kfS}} = \mathscr{L}_{\mathrm{NKA}}$

Eine Sprache L ist kontextfrei genau dann, wenn L von einem NKA akzeptiert wird.

Erzeugung eines äquivalenten NKA aus einer gegeben kfG $\quad kfG \rightarrow NKA$

Q = endliche Menge der Zustände
∑ = Eingabealphabet
Γ = Kelleralphabet
δ = partielle Überführungsfunktion
$\mathrm{q_0}$ = Anfangszustand, $\mathrm{q_0} \in \mathrm{Q}$
$\mathrm{k_0}$ = Kellervorbelegungszeichen, $\mathrm{k_0} \in \mathrm{T}$
E = Menge von Endzuständen, $\mathrm{E} \subseteq \mathrm{Q}$

$G = (N, T, P, s)$

$M = (\mathrm{Q}, \sum, \Gamma, \delta, \mathrm{q_0}, \mathrm{k_0}) = (\{\mathrm{q_0}\}, \mathrm{T}, \mathrm{N} \cup \mathrm{T}, \delta, \mathrm{q_0}, s)$

a) Für jede Regel $A \rightarrow \alpha \in P$ mit $\alpha \in (N \cup T)^*$ und $A \in N$ setze $\delta(\mathrm{q_0}, \varepsilon, A) \ni (q_0, \alpha)$

δ (Zustand, Eingabezeichen, oberstes Kellerzeichen) →(Zustand, Eingabezeichen im Keller)

b) Für alle Terminale $\alpha \in T$ setze $\delta(\mathrm{q_0}, a, a) \ni (q_0, \varepsilon)$. Das top of stack wird verbrauchend gelesen und muss mit dem aktuellen Zeichen auf dem Eingabeband übereinstimmen.

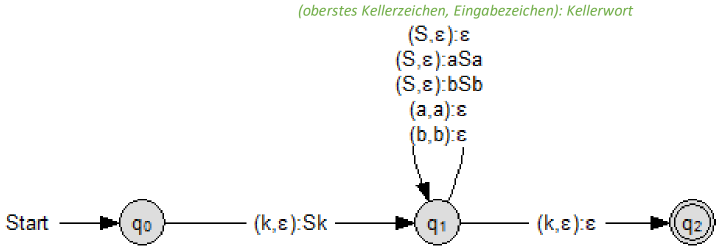
$G = (S, \{a, b\}, P, S)$

$M = (\{q_0, q_1, q_2\}, \{a, b\}, \{a, b, k, S\}, \delta, \mathrm{k}, \{\mathrm{q_2}\})$

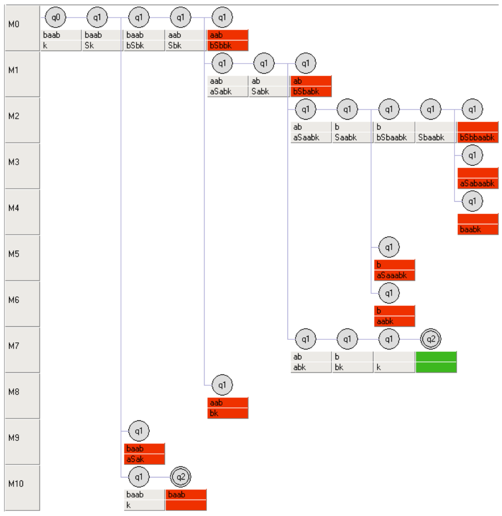
```
G = (N, T, P, s)

N = {S}
T = {a, b}
P = {
    S -> a S a | b S b | EPSILON
}
s = S
```

(oberstes Kellerzeichen, Eingabezeichen): Kellerwort



Zustand	Eingabe	Keller
q_0	`baab`	`k`
q_1	`baab`	`Sk`
q_1	`baab`	`bSbk`
q_1	`aab`	`Sbk`
q_1	`aab`	`aSabk`
q_1	`ab`	`Sabk`
q_1	`ab`	`abk`
q_1	`b`	`bk`
q_1	ε	`k`
q_2	ε	ε



Erzeugung eines äquivalenten kfG aus einer gegeben NKA

1. Fall: *reines Entkellern* bei $\delta(q, a, A) \ni (q', \varepsilon)$

Zustand	Eingabe	Keller
q	$a\beta$	$A\gamma$
q'	β	γ

In der „Welt der Grammatiken“ bedeutet das

$[q, A, q'] \longrightarrow a.$

- Alter Zustand q_i
- Oberster Kellerzeichen k_i
- Neuen Zustand q_j

$$[q_i, k_i, q_j]$$

2. Fall: *genau ein Kellerzeichen* bei $\delta(q, a, A) \ni (q_1, B)$

Zustand	Eingabe	Keller
q	$a\beta$	$A\gamma$
q_1		$B\gamma$
⋮		
q'	β	γ

In der „Welt der Grammatiken“ bedeutet das

$[q, A, q'] \longrightarrow a[q_1, B, q'].$

$[q, A, q']$ und $[q_1, A, q']$ sind Nichtterminale in G für alle $q' \in Q$.

3. Fall: Kellerwort der Länge 2 bei $\delta(q, a, A) \ni (q_1, BC)$

Zustand	Eingabe	Keller
q	$a\beta$	$A\gamma$
q_1		$BC\gamma$
⋮		
q_2		$C\gamma$
⋮		
q'	β	γ

In der „Welt der Grammatiken“ bedeutet das

$[q, A, q'] \rightarrow a[q_1, B, q_2]\, [q_2, C, q'].$

$[q, A, q']$, $[q_1, B, q_2]$ und $[q_2, C, q']$ sind Nichtterminale in G für alle q', $q_2 \in Q$.

$P = \{s \to [q_0, k_0, q] \mid$ für alle $q \in Q\}$

$\cup \{[q, A, q'] \to a \mid \delta(q, a, A) \ni (q', \varepsilon)\}$

$\cup \{[q, A, q'] \to a[q_1, B, q'] \mid \delta(q, a, A) \ni (q_1, B),$ für alle $q' \in Q\}$

$\cup \{[q, A, q'] \to a[q_1, B, q_2][q_2, C, q'] \mid \delta(q, a, A) \ni (q_1, BC),$ $\quad [q_0, Y, \square] \to 0[q_0, X, \diamond][\diamond, Y, \square]$

für alle $q', q_2 \in Q\}$ Für alle Zustandskombinationen

Beispiel:

$NKA\ M = (\{q_0, q_1\}, \{0,1\}, \{X, Y\}, \delta, q_0, Y)$

$\delta(q_0, 1, X) \ni (q_1, \varepsilon)$ $\quad [q_0, X, q_1] \to 1$

$\delta(q_1, 1, X) \ni (q_1, \varepsilon)$ $\quad [q_1, X, q_1] \to 1$

$\delta(q_1, \varepsilon, X) \ni (q_1, \varepsilon)$ $\quad [q_1, X, q_1] \to \varepsilon$

$\delta(q_1, \varepsilon, Y) \ni (q_1, \varepsilon)$ $\quad [q_1, Y, q_1] \to \varepsilon$

$[q_0, Y, \square] \to 0[q_0, X, \diamond][\diamond, Y, \square]$ Hilfestellung

$\delta(q_0, 0, Y) \ni (q_0, XY)$

$$[q_0, Y, q_0] \to 0[q_0, X, q_0][q_0, Y, q_0]$$
$$[q_0, Y, q_0] \to 0[q_0, X, q_1][q_1, Y, q_0]$$
$$[q_0, Y, q_1] \to 0[q_0, X, q_0][q_0, Y, q_1]$$
$$[q_0, Y, q_1] \to 0[q_0, X, q_1][q_1, Y, q_1]$$

$\delta(q_0, 0, X) \ni (q_0, XX)$

$$[q_0, X, q_0] \to 0[q_0, X, q_0][q_0, X, q_0]$$
$$[q_0, X, q_0] \to 0[q_0, X, q_1][q_1, X, q_0]$$
$$[q_0, X, q_1] \to 0[q_0, X, q_0][q_0, X, q_1]$$
$$[q_0, X, q_1] \to 0[q_0, X, q_1][q_1, X, q_1]$$

Nun werden alle Regeln entfernt, welche Nichtterminale enthält, wofür es keine Regel gibt. Oder ausschließlich rekursiv sind.

$[q_0, X, q_1] \to 1$

$[q_1, X, q_1] \to 1$

$[q_1, X, q_1] \to \varepsilon$

$[q_1, Y, q_1] \to \varepsilon$

$$\cancel{[q_0, Y, q_0] \to 0[q_0, X, q_0][q_0, Y, q_0]}$$
$$\cancel{[q_0, Y, q_0] \to 0[q_0, X, q_1][q_1, Y, q_0]}$$
$$\cancel{[q_0, Y, q_1] \to 0[q_0, X, q_0][q_0, Y, q_1]}$$
$$[q_0, Y, q_1] \to 0[q_0, X, q_1][q_1, Y, q_1]$$

$[q_1, Y, q_0]$ für dieses Nichtterminal existiert keine Regel.

$$\cancel{[q_0, X, q_0] \to 0[q_0, X, q_0][q_0, X, q_0]}$$
$$\cancel{[q_0, X, q_0] \to 0[q_0, X, q_1][q_1, X, q_0]}$$
$$\cancel{[q_0, X, q_1] \to 0[q_0, X, q_0][q_0, X, q_1]}$$
$$[q_0, X, q_1] \to 0[q_0, X, q_1][q_1, X, q_1]$$

Ausschließlich rekursiv

$[q_0, X, q_1] := A, \qquad [q_1, X, q_1] := B, \qquad [q_0, Y, q_1] := C$

$P = \{C \to 0A, A \to 0AB \mid 1, B \to 1 \mid \varepsilon\} \qquad G = (\{C, A, B\}, \{0,1\}, P, C)$

Deterministischer Kellerautomat $\mathscr{L}_{DKA} = \mathscr{L}_{dk\,fS} = \mathscr{L}_{LR(k)}$

$$M = (Q, \Sigma, \Gamma, \delta, q_0, k_0, E)$$

Ein DKA stoppt, wenn das Eingabewort vollständig abgetastet wurde und kein weiterer spontaner Zustandsübergang vorhanden ist. Falls es komplett abgetastet wurde und ein spontaner Übergang ist möglich, wird dieser aufjedenfall ausgeführt.

$Q =$ endliche Menge der Zustände
$\Sigma =$ Eingabealphabet
$\Gamma =$ Kelleralphabet
$\delta =$ partielle Überführungsfunktion
$q_0 =$ Anfangszustand, $q_0 \in Q$
$k_0 =$ Kellervorbelegungszeichen, $k_0 \in T$
$E =$ Menge von Endzuständen, $E \subseteq Q$

Der NKA $M_1 = (\{q_0, q_1\}, \{a; b\}, \{0, a, b\}, \delta, q_0, 0 \{q_1\})$ mit folgendem δ

δ (Zustand, Eingabezeichen, oberstes Kellerzeichen) →Zustand, Eingabezeichen im Keller

$\delta(q_0, a, b) \ni (q_0, \varepsilon)$

$\delta(q_0, a, 0) \ni (q_0, a0)$

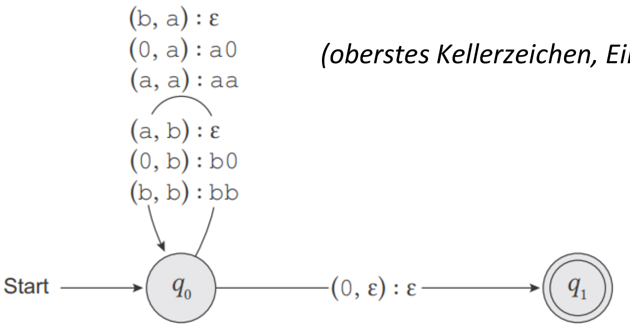
$\delta(q_0, a, a) \ni (q_0, aa)$

$\delta(q_0, b, a) \ni (q_0, \varepsilon)$

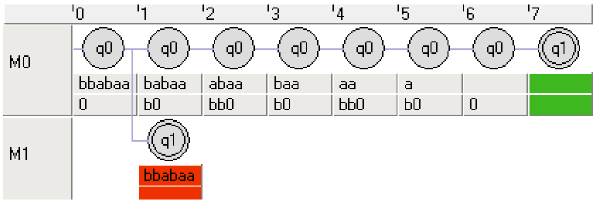
$\delta(q_0, b, 0) \ni (q_0, b0)$

$\delta(q_0, b, b) \ni (q_0, bb)$

$\delta(q_0, \varepsilon, 0) \ni (q_1, \varepsilon)$



(oberstes Kellerzeichen, Eingabezeichen): Kellerwort



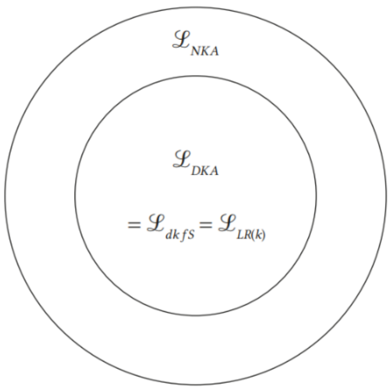
Zustand	Eingabezeichen	Keller (links: top of stack)	
q_0	bbabaa	0	Kellervorbelegungszeichen 0 **(Oberstes Kellerzeichen, Eingabezeichen): Kellerwort**
q_0	babaa	b0	(0, b): b0 → (q_0)
q_0	abaa	bb0	(b ,b): bb → (q_0)
q_0	baa	b0	(b, a): ε → (q_0)
q_0	aa	bb0	(b, b): bb → (q_0)
q_0	a	b0	(b, a): ε → (q_0)
q_0	ε	0	(b, a): ε → (q_0)
q_1	ε	ε	$(0, \varepsilon)$: ε → (q_1)

DKA vs. NKA

Für manche kfS kann ein NKA, jedoch kein äquivalenter DKA angegeben werden. $\mathscr{L}_{DKA} \subseteq \mathscr{L}_{NKA}$

Der DKA verfügt über keine Mittel, um festzustellen, ob die Wortmitte erreicht ist. Genau das ist die Voraussetzung, um den Vorgang des Kellerns zu beenden und auf Entkellern umzuschalten.

DKA beschreibt die Sprache der *deterministisch kontextfreien Sprachen* (*dkfS*). Sie bilden die für die Programmiersprachen wichtigste Sprachklasse und stimmen mit den sogenannten LR(k)-Sprachen überein.



Optimierung kontextfreier Grammatiken

Was für die Lesbarkeit und die Beherrschung der Komplexität des Entwurfsprozesses zuträglich ist, kann für den Prozess der Syntaxanalyse unnötigen Ballast bedeuten. Unnütze Nichtterminale führen beim praktischen Parsing zu vermeidbaren Zuständen und zeitaufwendigen Übergängen.

Unnütze Nichtterminale ausfiltern

a) Auf das Spitzensymbol s kann nicht verzichtet werden, da es Ausgangspunkt ist.
b) Für alle $X \in N$ mit $X \neq s\ gilt$
   1. X darf nicht gestrichen werden, wenn es mindestens ein Wort $w \in T^+$ gibt
   2. Wenn X in wenigstens einer von Spitzensymbol aus erreichbaren Satzform vorkommt.

$G = (N, T, P, s)$
$G = (\{A, B, C, D, E, F\}, \{a, b\}, P, A)$
$P = \{A \rightarrow aBb \mid aA \mid D, D \rightarrow E, E \rightarrow ab, B \rightarrow Fa, C \rightarrow abba\}$

$M_1 = \{X \mid X \in N\ und\ (X \rightarrow \alpha) \in P, mit\ \alpha \in T^+\}$ $\qquad$ $M1 = \{X \rightarrow T\}$ in P
$M_1 = \{E, C\}$

$M_{i+1} = M_i \cup \{X \mid X \in N\ und\ (X \rightarrow \alpha) \in P, mit\ \alpha \in (M_i \cup T)^+\}, i \geq 1$
$M_2 = M_1 \cup \{D \mid D \in N\ und\ (D \rightarrow \alpha) \in P, mit\ \alpha \in (M_1 \cup T)\text{^}+\} = \{E, C, D\}$
$M_3 = \{E, C, D, A\} = M_4$ Die Bildung der Mengen M_i wird bei Menge M_n beendet, wenn sich M_n nicht von M_{n+1} unterscheidet. Somit $n = 3$

Die ursprüngliche Grammatik G wird reduziert zu
$G' = (N', T', P', s') = (\{A, C, D, E\}, \{a, b\}, P', A)$
$P' = \{A \rightarrow aA \mid D, D \rightarrow E, E \rightarrow ab, C \rightarrow abba\}$

In P' werden nun alle Regeln $X \rightarrow \alpha$ gestrichen, die auf der rechten „tote Nichtterminale" Y enthalten, d.h. $Y \in (N \backslash M_n)$ und $Y \in \alpha$.
$M'_1 = \{s\} = \{A\}$
$M'_{i+1} = M'_i \cup \{X \mid X \in N'\ und\ (Z \rightarrow \alpha X ß) \in P'\ mit\ \alpha, ß \in (N' \cup T')^*, Z \in M'_i\}$
$M'_{i+1} = \mathrm{M}'_{\mathrm{i}} \cup \{X \mid X \in N'$ und $\mathrm{M}'_{\mathrm{i}} \rightarrow \alpha X ß \in P' \ldots\}$
$M'_2 = \{A, D\} = M_1 \cup \{D \mid D \in N'\ und\ (A \rightarrow \alpha D ß) \in P'\ mit\ \alpha, ß \in (N' \cup T')^*, A \in M'_i\}$
$M'_3 = \{\mathrm{A, D, E}\} = M_4$ $\qquad$ Somit $n = 3$
$G''(N'', T'', P'', s'') = (\{A, D, E\}, \{a, b\}, \{A \rightarrow aA \mid D, D \rightarrow E, E \rightarrow ab\}, A)$

Optimierung von kfG Kettenregeln

$G = (\{A, B, S\}, \{b\}, \{S \to A, A \to B, B \to b\}, S)$

a) Bilde die Menge M aller Paare $A, B \in NxN$ mit $A \Longrightarrow^* B$.

$M = \{(S, S), (S, A), (S, B), (A, A), (A, B), (B, B)\}$

b) Bilde die reduzierte Grammatik' $G' = (N, T, P', s)$

$P' = \{X \to ß \,|\, (X, Y) \in M\ und\ (Y \to ß) \in P, ß \notin N\}$

$P' = \{S \to b \,|\, (S, B) \in M\ und\ (Y \to b) \in P, ß \notin N\}$

$P' = \{S \to b, A \to b, B \to b\}$

Beispiel

$G = (\{A, D, E\}, \{a, b\}, \{A \to aA|D, D \to E, E \to ab\}, A)$

$M = \{(A, A), (A, D), (A, E), (D, D), (D, E), (E, E)\}$

$P' = A \to ab \,|\, (A, E) \in M\ und\ (E \to ab) \in P, ab \notin N\})$

$G = (\{A, D, E\}, \{a, b\}, \{A \to aA \,|\, ab, D \to ab, E \to ab\}, A)$

$M'_1 = \{A\}$

$G = (\{A\}, \{a, b\}, \{A \to aA|ab\}, A)$

Chomsky-Normalform

Eine kfG G=(N,T,P,s) ist in *Chomsky-Normalform*, kurz CNF, wenn jede Regel aus P entweder die Form $X \to a$ oder $X \to YZ$, mit $a \in T$ und $X, Y, Z \in N$ besitzt.

Zu jeder kfG G, mit $\varepsilon \notin L(G)$, gibt es eine äquivalente kfG G' in Chomsky-Normalform

1. Unnütze Nichtterminale ausfiltern
2. Kettenregeln beseitigen
3. abc. folgen.
   a. Übernimm alle Regeln der Form $X \to a$ mit $X \in N$ und $a \in T$ aus P in P'.
   b. Ersetzte in jeder Regel der Form $X \to \beta \in (N \cup T)^*$ und $|\beta| \geq 2$ jedes Terminal a durch X_a und ergänze die Regeln $X_a \to a$. X_a ist ein neues Nichtterminal. Übertrage die so veränderten Regeln in P'.
   c. Ersetze alle Regeln der $X \to Y_1 Y_2 \dots Y_n, n \geq 3$ durch

$$
\begin{aligned}
X &\to Y_1\ Z_1 \\
Z_1 &\to Y_2\ Z_2 \\
Z_2 &\to Y_3\ Z_3 \\
&\vdots \\
Z_{n-2} &\to Y_{n-1}\ Y_n
\end{aligned}
$$

Beispiel:

Wir betrachten die kfG $G = (\{S, A, B\}, \{a, b\}, P, s)$, mit $P = \{S \to aSa \mid aa \mid A, A \to bA \mid B, B \to b\}$.

1. Schritt: Kettenregeln eliminieren
Wir substituieren $X_1 \triangleq S$, $X_2 \triangleq A$, $X_3 \triangleq B$. In $\{X_1 \to aX_1a \mid aa \mid X_2, X_2 \to bX_2 \mid X_3, X_3 \to b\}$ streichen wir die Regel $X_2 \to X_3$ und ergänzen $X_2 \to b$. Dies führt zu $\{X_1 \to aX_1a \mid aa \mid X_2, X_2 \to bX_2 \mid b, X_3 \to b\}$.

Aus dieser Menge entfernen wir nun $X_1 \to X_2$ und ergänzen $X_1 \to bX_2 \mid b$. Das Ergebnis ist: $\{X_1 \to aX_1a \mid aa \mid bX_2 \mid b, X_2 \to bX_2 \mid b, X_3 \to b\}$. Durch Beseitigung unnützer Nichtterminale erhält man sogar $\{X_1 \to aX_1a \mid aa \mid bX_2 \mid b, X_2 \to bX_2 \mid b\}$.

2. Schritt: Neue Nichtterminale für Terminale ergänzen
$\{X_1 \to X_a X_1 X_a \mid X_a X_a \mid X_b X_2 \mid b, X_2 \to X_b X_2 \mid b, X_a \to a, X_b \to b\}$

3. Schritt: Reduzierung rechter Regelseiten
Nur die Regel $X_1 \to X_a X_1 X_a$ hat eine rechte Seite der Länge 3. Nach Substitution erhält man $\{X_1 \to X_a Z_1 \mid X_a X_a \mid X_b X_2 \mid b, Z_1 \to X_1 X_a, X_2 \to X_b X_2 \mid b, X_a \to a, X_b \to b\}$.

Das Ergebnis (nach Rücksubstitution der Nichtterminale) lautet: $G = (\{S, X_a, X_b, A, Z_1\}, \{a, b\}, \{S \to X_a Z_1 \mid X_a X_a \mid X_b A \mid b, Z_1 \to SX_a, A \to X_bA \mid b, X_a \to a, X_b \to b\}, S)$.

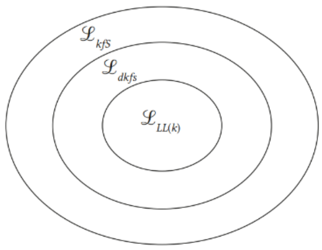
LL(k)-Sprachen

$\mathcal{L}_{LL(k)} \subset \mathcal{L}_{dkfS}$

Deterministisches Top-down-Parsing bedeutet, dass die in jedem Ableitungsschritt anzuwendende Regel eindeutig bestimmbar ist. Jede Regel erfolgt „irrtumsfrei", d.h. ohne Sackgassen, also ohne Backtracking. Die im jeweils nächsten Ableitungsschritt anzuwendende Produktion ist treffsicher *vorhersagbar*.

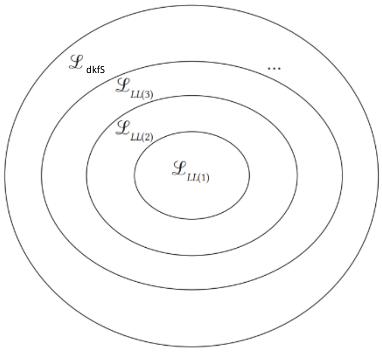
$LL(k) =$

- L=Analyse des Eingabewortes von links nach rechts
- L=Linksableitung des Analysewortes. Das am weitesten linksstehende Nichtterminal wird zuerst ersetzt.
- k=Anzahl der Vorausschauzeichen auf das noch nicht analysierte Rest Wort.



$LL(k)$-Grammatiken, die nur mit Vorausschau auf k-Folgezeichen auskommen, nennt man *stark-$LL(k)$-Sprachen.*

- Man kann zu jeder $LL(k)$-Grammatik eine äquivalente stark-$LL(k)$-Grammatik angeben.
- $LL(k)$-Grammatiken sind eindeutig. Mehrdeutige kfG können niemals LL(k)-Grammatik sein.
- Es gibt kfG, die für kein k $LL(k)$-Grammatik sind.
- Für eine gegebene kfS ist *nicht allgemein entscheidbar*, ob sie durch eine $LL(k)$-Grammatik definiert werden kann.
- Für eine gegebene kfS kann entschieden werden, ob ein festes k vom $LL(k) - Typ$ ist.



$\mathcal{L}_{LL(1)} \subset \mathcal{L}_{LL(2)} \subset \ldots \subset \mathcal{L}_{LL(i)} \subset \mathcal{L}_{LL(i+1)} \subset \ldots \subset \mathcal{L}_{dkfS}$

$LL(1)$-Forderungen 1

Eine Grammatik G erfüllt die *$LL(1)$-Forderung* 1, wenn für *jedes* Nichtterminal X von G, mit $X \longrightarrow \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n$ und $\alpha \in (N \cup T)^*$, gilt

$$FIRST(\alpha_i) \cap FIRST(\alpha_j) = \varnothing \quad \text{für alle} \quad i \neq j$$

Die ersten zu konstruierenden Zeichen dürfen niemals übereinstimmen.

Für eine Satzform α gilt:

$$FIRST(\alpha) := \{t \mid t \in T, \alpha \stackrel{*}{\Rightarrow}_G t\beta, \beta \in (N \cup T)^*\}$$

Falls $\alpha \stackrel{*}{\Rightarrow}_G \varepsilon$, gilt zusätzlich $FIRST(\alpha) \ni \varepsilon$. (Lies $\ni$ als „enthält".)

Beispiel: Es ist zu prüfen, ob die kfG G die $LL(1)$-Forderung erfüllt.
$G = (N, T, P, s) = (\{K, S, E\}, \{a, b, c, d\}, \{K \rightarrow S|\varepsilon, S \rightarrow aSb|E, E \rightarrow d|cE\}, K)$

$First(S)\ S \rightarrow \underline{a}Sb|E, E \rightarrow \underline{d}\ |\underline{c}E = \{a, d, c\}$

$S \rightarrow aSb$ - Es wird bis zum Nichtterminal geschaut $a \rightarrow t\beta$

Für K: $FIRST(S) = \{a, d, c\}, FIRST(\varepsilon) = \{\varepsilon\}$, $FIRST(S) \cap FIRST(\varepsilon) = \emptyset$
Für S: $FIRST(aSb) = \{a\}, FIRST(E) = \{d, c\}, FIRST(aSb) \cap FIRST(E) = \emptyset$
Für E: $FIRST(d) = \{d\}, FIRST(cE) = \{c\}, FIRST(d) \cap FIRST(cE) = \emptyset$

$LL(1)$-Forderungen 2

Eine Grammatik G erfüllt die $LL(1)$-Forderung 2, wenn für jedes Nichtterminal X, mit $X \overset{*}{\Rightarrow}_G \varepsilon$, gilt:

$$FIRST(X) \cap FOLLOW(X) = \emptyset$$

$$FOLLOW(X) := \{t \mid t \in T, s \overset{*}{\Rightarrow}_G \alpha X t \beta, \quad \text{mit} \quad \alpha, \beta \in (N \cup T)^*, X \in N\}$$

Eine kfG ist genau dann eine $LL(1)$-Grammatik, wenn sie beide Forderungen erfüllt.

Eine $LL(1)$-Grammatik kann niemals linksrekursiv oder mehrdeutig sein.

Für sämtliche Regeln der Form $A \longrightarrow \alpha_1 X \beta_1 \mid \alpha_2 X \beta_2 \mid \dots \mid \alpha_n X \beta_n$ mit $\alpha_i, \beta_i \in (N \cup T)^*$ ist $FOLLOW(X)$ wie folgt definiert:

$$FOLLOW(X) = \begin{cases} \bigcup_{i=1}^{n} FIRST(\beta_i), & \text{falls} \quad \varepsilon \notin \bigcup_{i=1}^{n} FIRST(\beta_i) \\ \left(\bigcup_{i=1}^{n} FIRST(\beta_i) \setminus \{\varepsilon\} \right) \cup FOLLOW(A), & \text{sonst} \end{cases}$$

Beispiel: $G = (\{E, X, T, Y, F\}, \{+, *, (,), a\}, P, E)$
$P = \{E \to TX, X -> +TX|\varepsilon, T \to FYa, Y \to * FY|\varepsilon, F \to (E)|a\}$

Für E: $FIRST(TX) = \{(, a\}$ $\quad E \to T, T \to F, F \to (|a$
Für X: $FIRST(+TX) = \{+\}, FIRST(\varepsilon)$ $\quad \mathrm{FIRST}(+\mathrm{TX}) \cap \mathrm{FIRST}(\varepsilon) = \emptyset$
Für T: $FIRST(FYA) = \{(, a\}$
Für Y: $FIRST(* FY) = \{*\}, FIRST(\varepsilon) = \{\varepsilon\}$ $\quad \mathrm{FIRST}(* \mathrm{FY}) \cap \mathrm{FIRST}(\varepsilon) = \emptyset$
Für F: $FIRST\big((E)\big) = \{(\}, FIRST(a) = \{a\}$ $\quad \mathrm{FIRST}((\mathrm{E})) \cap \mathrm{FIRST}(a) = \emptyset$

$LL(1) Forderung\ 1\ erfüllt.$

$FIRST(X) = \{+, \varepsilon), \mathrm{FOLLOW(X)} = \{)\}$ $\quad \mathrm{FIRST(X)} \cap \mathrm{FOLLOW(X)} = \emptyset$
$FIRST(Y) = \{*, \varepsilon), \mathrm{FOLLOW(Y)} = \{a\}$ $\quad \mathrm{FIRST(Y)} \cap \mathrm{FOLLOW(Y)} = \emptyset$

$LL(1) Forderung\ 2\ erfüllt.$

Grammatiktransformation

Für kfG, die keine $LL(1)$-Grammatik sind, kann man durch geeignete Transformationen versuchen, die gewünschte Form herzustellen.

1. ε-Freiheit herstellen
2. Kettenregeln eliminieren
3. Beseitigung von Linksrekursiven Regeln

$$X \to Xa_1|Xa_2|\dots|Xa_n|ß_1|ß_2|\dots|ß_n$$

$$X \to ß_1X'|ß_2X'|\dots|ß_nX'$$
$$X' \to a_1X'|a_2X'|\dots|a_nX'|\varepsilon$$

Beispiel
$G = (\{A, S\}, \{a, b, c, d\}, \{S \to Aa|b, A \to Ac|Sd|\varepsilon\}, S)$
Zuerst muss ε Freiheit hergestellt werden
$G' = \{A, s\}, \{a, b, c, d\}, \{S \to Aa|b|a, A - Ac|Sd|c\}, S)$
Danach beseitigen wir die Linksrekursion
$G'' = (\{A, A', S\}, \{a, b, c, d\}, \{S \to Aa|b|a, A - SdA'|cA', A' \to cA'|\varepsilon\}, S)$
Nun kann wieder ε-Freiheit hergestellt werden
$G''' = (\{A, A', S\}, \{a, b, c, d\}, \{S \to Aa|b|a, A \to SdA'|Sd|cA'|c, A' \to cA'|c\}, S)$

$LR(k)$-Sprachen

- L=Analyse des Eingabewortes von links nach rechts
- Rechtsableitung des Analysewortes. Da es eine Bottom-up-Analyse handelt, muss man das Ganze „auf den Kopf stellen“, d.h., es ist eine Linksreduktion
- k=Anzahl der Vorausschauzeichen auf das noch nicht analysierte Rest Wort.

Eine kfG ist *LR(k)-Grammatik*, wenn für jede Satzform $\alpha\beta\gamma$ in einer Rechtsableitung, mit $\gamma \in T^*$, $\beta \in (N \cup T)^+$ und $\alpha \in (N \cup T)^*$, das Handle β durch Vorausschau auf die ersten k Zeichen von γ eindeutig identifiziert werden kann. Es gilt $k \geq 0$.

Zu jeder $LR(k)$-Grammatik, mit $k > 1$, gibt es eine äquivalente $LR(1)$-Grammatik.

shift bedeutet, dass das nächste Zeichen (Token) aus dem Eingabepuffer (Restwort) entfernt und auf den Stapel gelegt wird.

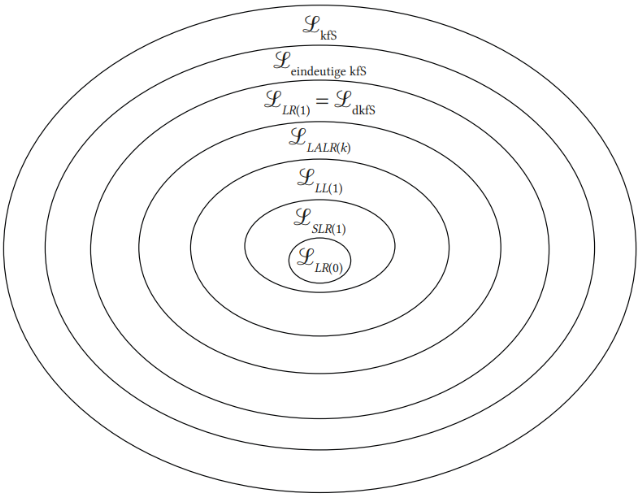
reduce: $X \longrightarrow \beta$ bedeutet, dass der Stapelinhalt gemäß Regel $X \longrightarrow \beta$ reduziert wird, wenn die rechte Regelseite $\beta \in (N \cup T)^+$ mit dem obersten Stapel(teil)wort übereinstimmt. Genau dieser Stapelinhalt wird durch die linke Regelseite, also X, ersetzt.

accepted steht ganz am Ende, wenn das Startsymbol der Grammatik als einziges Zeichen im Keller steht und der Puffer für das Eingabewort leer ist. Um das Ende des Eingabewortes zu kennzeichnen, verwenden wir ein Dollarzeichen \$.

$G = (\{E, T, F\}, \{+, *, a, (,)\}, \{E \to E + T \mid T, T \to T * F \mid F, F -> (E) \mid a\}, E)$

Stapel $\overset{\text{tos}}{\downarrow}$	Restwort	Aktion
$\$$	$a + a^*(a + a)\$$	$shift$
$\$\boldsymbol{a}$	$+a^*(a + a)\$$	$reduce: F \to a$
$\$\boldsymbol{F}$	$+a^*(a + a)\$$	$reduce: T \to F$
$\$\boldsymbol{T}$	$+a^*(a + a)\$$	$reduce: E \to T$
$\$\boldsymbol{E}$	$+a^*(a + a)\$$	$shift$
$\$\boldsymbol{E} +$	$a^*(a + a)\$$	$shift$
$\$\boldsymbol{E} + \boldsymbol{a}$	$*\ (a + a)\$$	$reduce: F \to a$
$\$\boldsymbol{E} + \boldsymbol{F}$	$*\ (a + a)\$$	$reduce: T \to F$
$\$\boldsymbol{E} + \boldsymbol{T}$	$*\ (a + a)\$$	$\boldsymbol{shift}$!!!
$\$\boldsymbol{E} + \boldsymbol{T}^*$	$(a + a)\$$	$shift$
$\$\boldsymbol{E} + \boldsymbol{T}^*($	$a + a)\$$	$shift$
$\$\boldsymbol{E} + \boldsymbol{T}^*(\boldsymbol{a}$	$+a)\$$	$reduce: F \to a$
$\$\boldsymbol{E} + \boldsymbol{T}^*(\boldsymbol{F}$	$+a)\$$	$reduce: T \to F$
$\$\boldsymbol{E} + \boldsymbol{T}^*(\boldsymbol{T}$	$+a)\$$	$reduce: E \to T$
$\$\boldsymbol{E} + \boldsymbol{T}^*(\boldsymbol{E}$	$+a)\$$	$shift$
$\$\boldsymbol{E} + \boldsymbol{T}^*(\boldsymbol{E} +$	$a)\$$	$shift$
$\$\boldsymbol{E} + \boldsymbol{T}^*(\boldsymbol{E} + \boldsymbol{a}$	$)\$$	$reduce: F \to a$
$\$\boldsymbol{E} + \boldsymbol{T}^*(\boldsymbol{E} + \boldsymbol{F}$	$)\$$	$reduce: T \to F$
$\$\boldsymbol{E} + \boldsymbol{T}^*(\boldsymbol{E} + \boldsymbol{T}$	$)\$$	$reduce: E \to E + T$
$\$\boldsymbol{E} + \boldsymbol{T}^*(\boldsymbol{E}$	$)\$$	$shift$
$\$\boldsymbol{E} + \boldsymbol{T}^*(\boldsymbol{E})$	$\$$	$reduce: F \to (E)$
$\$\boldsymbol{E} + \boldsymbol{T}^*\boldsymbol{F}$	$\$$	$reduce: E \to E + T$
$\$\boldsymbol{E}$	$\$$	$accepted$

Aus a wird F siehe Stapel



LR(0)-Sprachen	sind einfach zu analysieren, aber nicht mächtig genug, um alle gängigen Konstrukte in (imperativen) Programmiersprachen zu beschreiben.
LR(1)-Parsing,	kurz: *LR*-Parsing, ist die leistungsfähigste und in der Praxis am häufigsten benutzte Analysetechnik.
LR(k)-Parsing,	$k > 1$, ist praktisch wegen Satz 3.1 ohne Interesse.
	Jede *LL*(k)-Sprache ist auch *LR*(k)-Sprache.

Turing Maschine (TM) und Chomsky-Typ-0/1-Sprachen

Für Chomsky-Typ-0 Sprachen reichen NKA nicht aus. Z.B. gibt es keinen NKA, der diese Sprache $L = \{a^n b^n c^n | \ n \geq 0\}$ akzeptiert.

Eine Tuning Maschine **stoppt** und der erreichte Zustand ist *ein Endzustand.*
Eine Tuning Maschine **stoppt** und der erreichte Zustand ist k*ein Endzustand.*
Eine Turing Maschine stoppt nicht und läuft endlos weiter, dafür gibt es drei Realisierungsmöglichkeiten:
Unendlicher Rechtslauf. Unendlicher Linkslauf. Endloszyklus, der Kopf begebt sich über eine feste Anzahl von Feldern (auch null) hin und her. Wobei die zum Zyklus gehörende „Taktlänge“ keine Rolle spielt.

Eine DTM **stoppt per *crash*** bedeutet, dass der nächste Schritt nicht definiert ist.
Sprich, Zu dem Zeichen, welches in diesem Zustand eingelesen wurde, gibt es keine Überführungsfunktion.

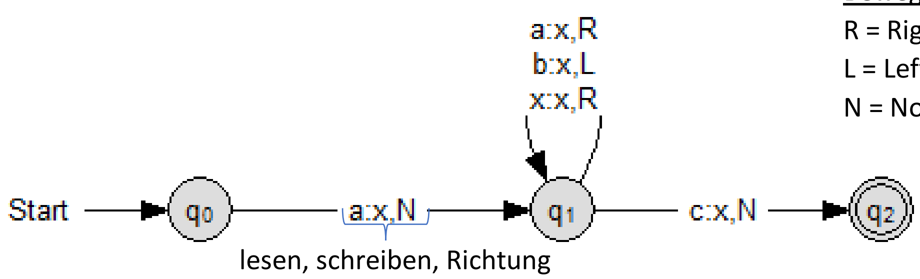
Eine deterministische Turing Maschine (DTM) ist ein 7-Tupel $M = (Q, \sum, \Gamma, \delta, q_0, \$, \mathrm{E})$.
$Q = endliche\ Menge\ von\ Zuständen$
$\sum = \text{Eingabealphabet}$
$\Gamma = \text{Bandalphabet, wobei } \sum \subseteq \mathrm{T}\{\$\}$
$\delta = \text{partielle Überführungsfunktion: } \mathrm{Q} * \mathrm{t} * \{\mathrm{L}, \mathrm{N}, \mathrm{R}\};$
$q_0 = Anfangszustand, q_0 \in Q$
$\$ = Bandvorbelegungszeichen, kurz{:}\ Blankzeichen, \$ \in T \backslash \sum$
$E = endliche\ Menge\ von\ Endzuständen, mit\ E \subseteq \mathrm{Q}$

Wir betrachten die folgende sehr einfache DTM
$M_1 = (\{q_0, q_1, q_2\}, \{a, b, c\}, \{\$, a, b, c, x\}, \delta, q_0, \$, \{q_2\})$ mit

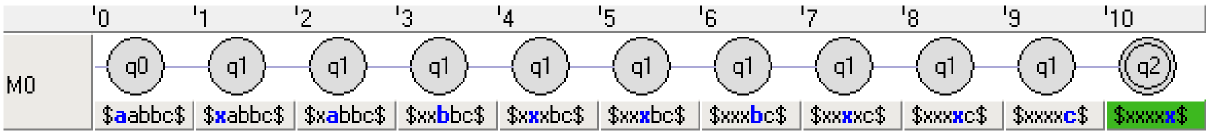
δ	$	*a*	*b*	*c*	*x*
q_0	–	(q_1, x, R)	–	–	–
q_1	$(q_1, \$, N)$	(q_1, x, R)	–	(q_2, x, L)	–
q_2	–	–	–	–	–

Die mit einem Strich markierten Tabellenfelder bedeuten, dass die Überführungsfunktion δ für die zugehörigen Argumente nicht definiert ist.

Die Überführungsfunktion kann auch als Graph dargestellt werden:



<u>Bewegungsrichtung</u>
R = Right
L = Left
N = None



DTM als Akzeptor

Ein Wort *w* wird akzeptier, wenn die auf w angesetzte DTM in endlich vielen Schritten *in einem Endzustand* stoppt. Dabei ist es völlig gleichgültig, was danach auf dem Band steht, wo sich der Lese-Schreib-Kopf befindet und ob das Wort überhaupt ganz gelesen wurde.
Es ist möglich, dass lediglich das erste Zeichen vom Wort eingelesen wurde und der DTM das Wort akzeptiert, dies steht in strengem Gegensatz zu endlichen und Kellerautomaten, die das gesamte Eingabewort stets vollständig abtasten.

Kontextsensitive Sprachen

Eine Sprache L heißt *rekursiv* (*entscheidbar*), wenn es eine stets anhaltende DTM M gibt, mit $L = L(M)$.

Man kann zeigen, dass die Menge aller rekursiven (algorithmisch erkennbaren) Sprachen $\mathscr{L}_{\mathrm{r}}$ eine Untermenge der Menge der rekursiv aufzählbaren Sprachen $\mathscr{L}_{\mathrm{re}}$ ist, die Klasse der ksS jedoch vollständig enthält. „re" steht für „recursive enumerable".

$$\mathscr{L}_{\mathrm{Typ\,1}} \subset \mathscr{L}_{\mathrm{r}} \subset \mathscr{L}_{\mathrm{re}}$$

Die Klasse der Chomsky-Typ-1-Sprachen wird durch *linear beschränkte TM*, kurz: LBTM, beschrieben, d.h.:

$$\mathscr{L}_{\mathrm{LBTM}} = \mathscr{L}_{\mathrm{Typ1}}$$

Überabzählbar unendliche Mengen
Bsp.: Menge aller Wortmengen über einem Alphabet
- **Abzählbare Mengen**
  Bsp.: Menge der wahren arithmetischen Formeln
  - **rekursiv aufzählbare Spr. = DTM/NTM-Spr. = Chomsky-Typ-0-Spr.**
    Bsp.: Menge aller Algorithmen, Halteproblem
    - **rekursive Sprachen = Sprachen der stets stoppenden TM**
      - **ksS = LBTM-Sprachen = Chomsky-Typ-1-Sprachen**
        Bsp.: $L = \{a^n b^n c^n \mid n \in N\}$
        - **kfS = NKA-Sprachen = Chomsky-Typ-2-Sprachen**
          Bsp.: $L = \{a^n b^n \mid n \in N\}$
          - **dkfS = DKA-Sprachen**
            Bsp.: Palindrome mit markierter Wortmitte
            - **rS = regExp-Sprachen = Chomsky-Typ-3-Sprachen**
              Bsp.: Zahlwörter
              - **endliche Sprachen (alle Elemente angebbar)**

Abb. 4.3: Hierarchie von Sprachklassen (ohne *LR*- und *LL*-Sprachen)