Anwendung (RA):
man grep
man sed
vi / vim
emacs
flex / lex

1.4. Kontextfreie Sprachen (Kfs) und Kellerautomaten (KA)

Ch-3-G genügen i.a. nicht um komplexere formale Sprachen wie Programmiersprachen zu beschreiben (s. z.B. Klammerstrukturen wie in $Lab = \{a^n b^n : n \in \mathbb{N}\}$).
Es gilt $REG_G \subsetneq CFL$. D.h. die Klasse der Kfs ist mächtiger als die der regulären Sprachen (rs).
CH-3-G, RA, und EA eignen sich nicht um Kfs zu beschreiben.
Analog zu RA, EA und Ch-3-G ~~gibt es~~ bei rS sind in der Klasse Kfs folgende Beschreibungsformen gleich mächtig:

- Ch-2-G
- BNF-Ausdrücke
- Kellerautomaten

Diese Mittel werden bei Spezifikation und der Syntaxanalyse von Programmiersprachen in Übersetzern und Interpretierern eingesetzt.

1.4.1. BNF-Notation

Erweiterung von RA:
1. Hilfszeichen (Nichtterminale)
2. (rekursive) Gleichungen für Hilfszeichen.

Sei T eine Menge von Terminalzeichen und N eine Menge von Nichtterminalzeichen. Eine BNF-Beschreibung einer Sprache L hat die Form:

$$X_1 ::= E_1$$
$$\vdots$$
$$X_n ::= E_n \quad \text{mit } X_1, \dots, X_n \in N \text{ und } E_1, \dots, E_n \text{ RA über } T \cup N.$$

Bsp $\langle Z \rangle ::= ab \mid a\langle Z \rangle b$ beschreibt die Kfs $\{a^n b^n : n \in \mathbb{N} \setminus \{0\}\}$

Für eine BNF-Beschreibung kann für jedes X_i $(1 \le i \le n)$ eine Ch-2-G (KfG) angegeben werden. Die RA E_i werden in Ch-3-G (mit neuem Nichtterminal Z als Axiom) umgeformt und für die BNF-Regel

$$X_i ::= E_i$$

wird die Regel $Z_i \to x_i$ in die Grammatik aufgenommen. □

Anmerkung: Es gibt zahlreiche Stile und Erweiterungen von BNF.
Die Klammern [...] dienen oft der Kennzeichnung optionaler Anteile; z.B. $X[Y]$ für $X \mid XY$. ($X[Y]$ sprich X optional Y)
$\{X\}_n^m$ bezeichnet die n- bis m-malige Wiederholung von X.

1.4.2. Kellerautomaten (KA)

Ein Kellerautomat $KA = (S, T, K, \delta, s_0, k_0, S_z)$
- S endl. Menge von Zuständen
- T " von Eingabezeichen
- K " von Kellerzeichen
- δ endl. Übergangsfunktion $\delta: S \times (T \cup \{\varepsilon\}) \times K \to \mathcal{P}(S \times K^*)$
- $s_0 \in S$ Anfangszustand
- $S_z \subseteq S$ Endzustände

Prinzip: KA verarbeitet ein Wort $w \in T^*$, indem er w von links nach rechts liest und in jedem Schritt abhg. vom aktuellen Zustand s Eingabezeichen

Prinzip: KA verarbeitet ein Wort $w \in T^*$, indem er w von links nach rechts liest, und in jedem Schritt abhg vom aktuellen Zustand s, Eingabezeichen a und obersten Kellerzeichen k, entsprechend δ folgendes tut:

- in einen neuen Zustand $s' \in S$ übergeht
- das oberste Zeichen k im Keller entfernt
- eine (ggf. leere) Sequenz $u \in K^*$ auf den Keller legt

KA akzeptiert $w \in T^*$, wenn die vollst. Verarbeitung von w von s_0 zu einem Endzustand $s \in S_z$ führt.

Darstellung Graph analog zu EA. Knoten: $s \in S$, ~~Kanten (a,k,u)~~
Kanten: $(a,k,u) \in (\Gamma \cup \{\varepsilon\}) \times K \times K^*$.
Eine Kante (a,k,u) von s nach s' existiert g.d.w. $(s',u) \in \delta(s,a,k)$

Bsp. (KA für $\{(^n)^n : n \in \mathbb{N} \setminus \{0\}\}$)

$KA = (S, T, K, \delta, s_0, 0, \{s_z\})$

$S = \{s_0, s_1, s_2\}$ $\quad T = \{(,)\}$ $\quad K = \{1, 0\}$

Übergangsrelation δ:

$\delta(s_0, (, 0) = \{(s_0, <10>)\}$ ← eine öffnende Klammer gelesen → auf den Keller ablegen

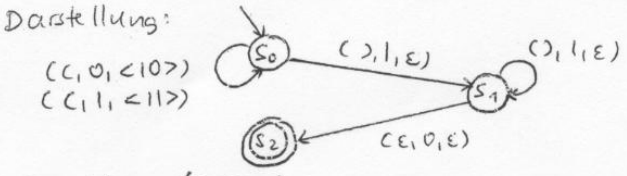
$\delta(s_0, (, 1) = \{(s_0, <11>)\}$ → auf den Keller ablegen

$\delta(s_0,), 1) = \{(s_1, \varepsilon)\}$ $\quad s_1$: Zustand = Erkennung von schließenden Klammern

$\delta(s_1,), 1) = \{(s_1, \varepsilon)\}$ → nichts auf dem Keller ablegen

$\delta(s_1, \varepsilon, 0) = \{(s_2, \varepsilon)\}$

Darstellung:



Arbeitsweise formal:
wir betrachten Kellerkonfiguration $(s, w, v) \in S \times T^* \times K^*$ mit Kontrollzustand s, (Rest-) Eingabewort w und Kellerzwtand v.
Ein KA induziert eine Übergangsrelation $\rightarrow$ auf Konfigurationen, vermöge: $(s_1, w, <k> \circ v) \rightarrow (s_2, w, u \circ v)$ falls $(s_2, u) \in \delta(s_1, \varepsilon, k)$
und $(s_1, <a> \circ w, <k> \circ v) \rightarrow (s_2, w, u \circ v)$ falls $(s_2, u) \in \delta(s_1, a, k)$

$w \in T^*$ von KA akzeptiert, wenn es ein $s \in S_z$ und ein $v \in K^*$ gibt, so dass: $(s_0, w, <k_0>) \xrightarrow{*} (s, \varepsilon, v)$ (Akzeptanz durch Endzustände)

Die von KA akzeptierte Sprache bezeichnet wir mit $L(KA)$.

Anm. Akzeptieren durch „leeren Keller" ist gleich mächtig.
D.h. $w \in T^*$ von KA akzeptiert, wenn $(s_0, w, \varepsilon) \xrightarrow{*} (s, \varepsilon, \varepsilon)$ mit $s \in S$.

Durch KA erhalten wir für beliebige kfS unmittelbar einen (i.d.R. sehr effizienten) Erkennungsalgorithmus!
Im Gegensatz zu EA sind deterministische KA nicht äquivalent zu nichtdeterministischen KA!

1.4.3. Äquivalenz von KA und Kfs

Satz: Ist L eine Kfs, dann gibt es einen KA K, so dass $L(K) = L$.

Idee: Geg. sei eine Kfg $G = (T, N, \to, Z)$.
Ges. KA K mit $L(K) = L(G)$
$K = (S, \Gamma, \Gamma \cup N \cup \{\#\}, \delta, \varepsilon, \#, \{s_e\})$
(Keller-Ende-Zeichen)

Wähle $S = \{\varepsilon, s_e, s_v\} \cup$ Menge aller (Teil-)sequenzen der linken Seiten der Grammatik-Regeln.
Sei $\langle e_1, \dots, e_n \rangle$, $e_i \in N \cup T$, linke Seite einer Grammatik-Regel
$\langle e_1, \dots, e_n \rangle \to A$ (A = rechte Seite d. Regel)

Übergangsrelation δ:

1. Kellern $(\varepsilon, \langle a k \rangle) \in \delta(\varepsilon, a, k)$ $\quad a \in T$
2. Regelerkennung:
   a) $(\langle e_{i-1} \dots e_j \rangle, \varepsilon) \in \delta(\langle e_i \dots e_j \rangle, \varepsilon, e_{i-1})$ (Teilsequenz einer linken Seite) — ein Zeichen vom Keller holen und „vorne" ran tun
   b) $(\langle e_i \dots e_{j+1}, \langle k \rangle) \in \delta(\langle e_i \dots e_j \rangle, e_{j+1}, k)$
3. Regelanwendung:
   $(\varepsilon, \langle A k \rangle) \in \delta(\langle e_1, \dots, e_n \rangle, \varepsilon, k)$
4. Akzeptanz:
   a) $(s_v, \varepsilon) \in \delta(\varepsilon, \varepsilon, z)$ (Zust. / Eingabe / Keller)
   b) $(s_e, \varepsilon) \in \delta(s_v, \varepsilon, \#)$

Automat muss an d. richtigen Stelle mit Regelaufbau beginnen

K ist nichtdeterministisch und ineffizient!
K kellert Eingabe, bis er an bel. Stelle mit dem Aufbau der linken Seite einer Regel beginnt.

Beispiel: Kfg $G = (\{a, b\}, \{z\}, \{azb \to z, zz \to b, ab \to z\})$

Konstruiere Kellerautomaten:

$S = \{\varepsilon, s_v, s_e, \langle a \rangle, \langle b \rangle, \langle z \rangle, \langle az \rangle, \langle zb \rangle, \langle zz \rangle, \langle azb \rangle, \langle ab \rangle\}$
δ: (sei $X \in T$, $k \in K$) $\quad \delta(\varepsilon, x, k) = \{(\langle x \rangle, \langle k \rangle), (\varepsilon, \langle x k \rangle)\}$

$k \neq \# \Rightarrow \delta(\varepsilon, \varepsilon, k) = \{(\langle k \rangle, \varepsilon)\}$

2a $\quad \delta(\langle z \rangle, b, k) = \{(\langle zb \rangle, \langle k \rangle), (\langle zz \rangle, \langle k \rangle)\}$

$\delta(\varepsilon, \varepsilon, z) = \{(s_v, \varepsilon)\}$

$\vdots$

Ausschnitt aus Ableitungsbaum für $\langle aabb \rangle$:

$(\varepsilon, aabb, \#) \xrightarrow{1} (\varepsilon, abb, a\#) \xrightarrow{1} (\varepsilon, bb, aa\#) \xrightarrow{1} (\varepsilon, b, baa\#) \to (\varepsilon, \varepsilon, bbaa\#)$

$(a, bb, a\#)$ (2b) $\quad (b, \varepsilon, baa\#)$ (2a)

$(ab, b, a\#)$ (2b)

3

$(\varepsilon, b, za\#) \xrightarrow[2b\ 2a\ 2a]{2a\ 2a\ 2b} (azb, \varepsilon, \#)$

3

$(\varepsilon, \varepsilon, z\#)$

4a 4b

$(s_e, \varepsilon, \#)$ ✓ wort akzeptiert.

Beobachtung: K nichtdeterministisch und sackgassen d.h. sehr ineffizient. KA akzeptiert selbe Sp.

Bei [illegible] der Form $\langle e_1 \dots e_m \rangle \to A$ für $w = x \circ \langle e_1 \dots e_m \rangle \circ y$ durch Betrachtung eines endlichen Präfix von y gesteuert werden.

Kontextbedingung (KB): Menge von Wörtern, die die Präfixe von y festlegt, welche die Regelanwendung zulassen.

KFG G heißt LR-deterministisch, wenn es für alle Regeln endliche KB gibt, so dass der von links nach rechts arbeitende KA für alle $w \in L(G)$ und alle akzeptierenden Reduktionen in jedem Schritt genau eine Regel zulässt, die Reduktion vollzieht und Sackgassen vermeidet.

Satz: Jede LR-deterministische KFG ist eindeutig.

Beweis: gemäß Definition LR-deterministischer KFG gibt es eine eindeutige Einschränkung des KA, der die Ableitung erzeugt.

Def. (LR(k))

$w[i:k]$ bezeichnet Teilwort $\langle w_i \dots w_k \rangle$ von $w = \langle w_1 \dots w_n \rangle$ $\quad i, k \in \mathbb{N}, 1 \le i$

Eine azyklische KFG $(T, N, \to, Z)$ heißt LR(k)-Grammatik, wenn für jedes Paar von Ableitungen in Linksnormalform gilt:

$u_1 \circ a_1 \circ x_1 \Longrightarrow u_1 \circ \langle \beta_1 \rangle \circ x_1 \Longrightarrow \dots \Longrightarrow z$ $\quad$ ⟹ Regelanwendung

$u_2 \circ a_2 \circ x_2 \Longrightarrow u_2 \circ \langle \beta_2 \rangle \circ x_2 \Longrightarrow \dots \Longrightarrow z$

wobei $\{a_1 \to \beta_1, a_2 \to \beta_2\} \subseteq \to$ gilt:

$u_1 \circ a_1 \circ (x_1[1:k]) = u_2 \circ a_2 \circ (x_2[1:k]) \Rightarrow a_1 = a_2 \wedge \beta_1 = \beta_2 \wedge u_1 = u_2$ □

D.h., durch $x_1[1:k]$ ist die anzuwendende Regel eindeutig festgelegt.

Bsp. (LR(0))

$G = (\{a, b\}, \{Z, X\}, \{Zx \xrightarrow{1} Z,\ X \xrightarrow{2} Z,\ aZb \xrightarrow{3} X,\ ab \xrightarrow{4} X\}, Z)$

Ableitung von $\langle aabbab \rangle$: (Sprache $a^n b^n ab$)

$\langle a\underline{ab}bab \rangle \xrightarrow{4} \langle a\underline{X}bab \rangle \xrightarrow{2} \langle \underline{aZb}ab \rangle \xrightarrow{3} \langle \underline{X}ab \rangle \xrightarrow{2} \langle Z\underline{ab} \rangle \xrightarrow{4} \langle \underline{ZX} \rangle \xrightarrow{1} \langle Z \rangle$,

KA: kellern bis Regel anwendbar, Regeln anwenden so lange wie möglich.

Keller: Präfix des zu reduzierenden Wortes!

Keller	Restwort
ε	aabbab
a	abbab
aa	bbab
aab	bab
aX	bab
aZ	bab
⋮	⋮
Z	

Bei LR(0) ist kein Kontext notwendig, um die richtigen Regeln zu wählen. □

Satz: Jede LR(k)-Grammatik ist eindeutig.

Beweis: LR(k)-Grammatiken sind LR-deterministisch

Bsp. (LR(1)) $G = (T, N, \to, Z)$ $N = \{Z, A, P, E\}$ $T = \{(,), +, -, *, /, a\}$

Regel	Kontextbedingungen
$A \to Z$	ε
$E \to A$	$+, -,), \varepsilon$
$A+E \to A$	$+, -,), \varepsilon$
$A-E \to A$	$+, -,), \varepsilon$
$P \to E$	$t \in T$
$E*P \to E$	$t \in T$
$E/P \to E$	$t \in T$
$(A) \to P$	$t \in T$
$a \to P$	$t \in T$

(da auf der linken Seite nur ein A: $+, -,), \varepsilon$ folgt)

Bsp. $\langle a + a * a \rangle \Longrightarrow \dots \Longrightarrow \langle \underline{A+E} * \underline{a} \rangle \Longrightarrow$

$\langle A+E*P \rangle \Longrightarrow \dots Z$

LR(k)-Grammatiken erlauben Reduktion von Wörtern durch KA mit akzeptablem Aufwand. Insbesondere LR(1)-G werden in der Praxis eingesetzt.

Anmerkung: Es gibt keinen Algorithmus, der für eine beliebige Grammatik G entscheidet, ob G LR(k) ist. d.h. es kann sein, dass kein Alg. ex.

1.4.6. LL(k)-Grammatiken (Left - Leftmost)

Gegensatz zu LR(k): top-down Produktion der Ableitung startend von Axiom. Dadurch Linksableitung (bzw. Rechtsreduktion) statt Linksreduktion.

Idee (KA):

1. Produktion von $v \in T^*$ ausgehend von Nichtterminal auf Keller und abhängig vom Rechtskontext.
2. Vergleich des erzeugten v mit dem Präfix des Resteingabewortes.
3. Bei Übereinstimmung entfernen wir v sowohl aus dem Keller als auch aus dem Eingabewort.

Def (LL(k))

Eine azyklische, kfG $G = (T, N, \to, Z)$ heißt LL(k)- Gram. ($k \in \mathbb{N}$), falls für alle Paare von Ableitungen in Rechtsnormalform ($a_1, a_2, v, u_1, u_2 \in (T \cup N)^*$, $w \in T^*$, $\beta \in N$):

$v \circ u_1 \xrightarrow{+} v \circ a_1 \circ w \xrightarrow{*} v \circ \langle \beta \rangle \circ w \xrightarrow{*} z$
$v \circ u_2 \xrightarrow{*} v \circ a_2 \circ w \xrightarrow{*} v \circ \langle \beta \rangle \circ w \xrightarrow{*} z$

} wobei $\{a_1 \to \beta, a_2 \to \beta\} \subseteq \to$ gilt: $u_1[1:k] = u_2[1:k] \longrightarrow a_1 = a_2$.

D.h. die Zerteilung ist mit $u_1[1:k]$ eindeutig festgelegt.