1. Chomsky Hierarchie

1.1. Definition:

(i) Eine *Grammatik* $G = (V, \Sigma, P, S)$ ist ein 4-Tupel mit — *Grammatik*

V: endliche Menge von *Variablen* — *Variable*

Σ: endliche Menge von *Terminalsymbolen* — *Terminalsymbol*

S: *Startsymbol* — *Startsymbol*

P: $\subseteq ((V \cup \Sigma)^* \setminus \Sigma^*) \times (V \cup \Sigma)^*$ Menge der *Produktionen.* — *Produktion*

Schreibweise: $\alpha \to \beta$ für $(\alpha, \beta) \in P$

(ii) Seien $G = (V, \Sigma, P, S)$ Grammatik, $u, v \in (V \cup \Sigma)^*$.
$u \to v :\Leftrightarrow \exists \alpha, \beta, x, y \in (V \cup \Sigma)^* : u = \alpha x \beta \wedge v = \alpha y \beta \wedge (x, y) \in P$

(iii) $\overset{*}{\to}$ ist der reflexive transitive Abschluss von $\to$

(iv) von der Grammatik G *erzeugte Sprache*: $L(G) := \{w \in \Sigma^*; S \overset{*}{\to} w\}$ — *erzeugte Sprache*

1.2. Definition (eingeschränkte Grammatiken): Sei $G = (V, \Sigma, P, S)$ Grammatik.

(i) G *rechtslinear* $:\Leftrightarrow \forall (\alpha \to \beta) \in P : \alpha \in V \wedge (\exists u \in \Sigma, B \in V : \beta = uB \vee \beta = u)$ — *rechtslinear*

(ii) G *linear* $:\Leftrightarrow \forall (\alpha \to \beta) \in P : \alpha \in V \wedge (\exists u, v \in \Sigma, B \in V : \beta = uBv \vee \beta = u)$ — *linear*

(iii) G *kontextfrei* $:\Leftrightarrow \forall (\alpha \to \beta) \in P : \alpha \in V \wedge \beta \in (V \cup \Sigma)^*$ — *kontextfrei*

(iv) G *kontextsensitiv* $:\Leftrightarrow \begin{cases} \forall (\alpha \to \beta) \in P : & \exists u, v, x \in (V \cup \Sigma)^*, A \in V : \alpha = uAv \\ & \wedge \beta = uxv \wedge |x| \geq 1 \end{cases}$ — *kontextsensitiv*

(v) G *beschränkt* $:\Leftrightarrow \forall (\alpha \to \beta) \in P : |\alpha| \leq |\beta|$ — *beschränkt*

(vi) G heißt allgemein, wenn keine Einschränkungen gelten

1.3. Lemma: Zu jeder beschränkten Grammatik G gibt es eine kontextsensitive Grammatik G' mit $L(G) = L(G')$.

Beweis: Sei $G = (V, \Sigma, P, S)$ eine beschränkte Grammatik. Konstruktion von $G' = (V', \Sigma, P', S)$ kontextsensitiv mit $L(G) = L(G')$. Sei $(\alpha \to \beta) \in P$. Betrachte folgende Fälle:

(i) $|\alpha| = 1$: Regel ist kontextsensitiv und wird in P' aufgenommen

(ii) $|\alpha| = 2$: $X_1 X_2 \to Y_1 Y_2 \ldots V_r$ mit $r \geq 2$, $X_1, X_2, Y_1, \ldots, Y_r \in V \cup \Sigma$, Einführung von zwei neuen Variablen: $H_1, H_2 \in V' \setminus V$ und Regeln $X_1 X_2 \to H_1 X_2$, $H_1 X_2 \to H_1 H_2 Y_3 \ldots Y_r$, $r \geq m$, $H_1 H_2 \to Y_1 H_2$, $Y_1 H_2 \to Y_1 Y_2$

(iii) $|\alpha| = m > 2$: $X_1 \dots X_m \to Y_1 \dots Y_r$, Einführung von m neuen Variablen $H_1, \dots, H_m \in V' \setminus V$ und Regeln $X_1 \dots X_m \to H_1 X_2 \dots X_m$, $H_1 X_2 \dots X_m \to H_1 H_2 X_3 \dots X_m$, $\dots$, $H_1 \dots H_{m-1} X_m \to H_1 \dots H_m Y_{m+1} \dots Y_r$, $H_1 \dots H_m \to Y_1 H_2 \dots H_m$, $\dots$, $Y_1 \to Y_{m-1} H_m \to H_1 \dots H_m$ (insgesamt $2m$ Regeln).

□

1.4. Definition (Sprachklassen):

reguläre Sprachen
(i) $\mathcal{L}_3 = \mathrm{REG} = \{L(G) | G \text{ rechtslinear}\}$ Klasse der *regulären Sprachen*

lineare Sprachen
(ii) $\mathrm{LIN} = \{L(G) | G \text{ linear}\}$ Klasse der *linearen Sprachen*

kontextfreie Sprachen
(iii) $\mathcal{L}_2 = \mathrm{CFL} = \{L(G) | G \text{ kontextfrei}\}$ Klasse der *kontextfreien Sprachen*

kontextsensitive Sprachen
(iv) $\mathcal{L}_1 = \mathrm{CSL} = \{L(G) | G \text{ kontextsensitiv}\}$ Klasse der *kontextsensitiven Sprachen*

allgemeine Sprachen
(v) $\mathcal{L}_0 = \{L(G) | G \text{ allgeine Grammatik}\}$ Klasse der *allgemeinen Sprachen*

1.5. Bemerkung (Beziehungen zwischen Sprachklassen): Es gilt:

$$\mathcal{L}_3 \subsetneq \mathrm{LIN} \subsetneq \mathcal{L}_2 \subsetneq \mathcal{L}_1 \subsetneq \mathcal{L}_0$$

Beweis: Teilmengenbeziehung klar, Ungleichheit:

(i) $\mathcal{L}_3 \neq \mathrm{LIN}$: $\{a^n b^n | n \in \mathbb{N}\} \in \mathcal{L}_3 \setminus \mathrm{LIN}$

(ii) $\mathrm{LIN} \neq \mathcal{L}_2$: $\{a^n b^n c^m d^m | m, n \in \mathbb{N}\} \in \mathcal{L}_2 \setminus \mathrm{LIN}$

(iii) $\mathcal{L}_2 \neq \mathcal{L}_1$: $\{a^n b^n c^n | n \in \mathbb{N}\} \in \mathcal{L}_1 \setminus \mathcal{L}_2$

(iv) $\mathcal{L}_1 \neq \mathcal{L}_0$: Beweis s.u. (Abschlußeigenschaft)

□

1.6. Definition: Seien X, Y Alphabete und $h : X \to Y^*$ eine Funktion.

Homomorphismus
(i) Die eindeutige Erweiterung $h : X^* \to Y^* : h(a_1 \dots a_n) := h(a_1) \dots h(a_n)$ heißt *Homomorphismus*.

(ii) Sei $L \subseteq X^*$. $h(L) := \{h(v) | v \in L\}$

(iii) Sei $L \subseteq Y^*$. $h^{-1}(L) := \{w \in X^* | h(w) \in L\}$

(iv) Sei $\mathcal{L}$ eine Menge von Sprachen. $\mathrm{HOM}(\mathcal{L}) := \{h(L) | L \in \mathcal{L}, h \text{ Homomorphismus}\}$ und $\mathrm{HOM}^{-1}(\mathcal{L}) := \{h^{-1}(L) | L \in \mathcal{L}, h \text{ Homomorphismus}\}$

1.7. Definition: Eine Sprachklasse $\mathcal{L}$ heißt *abgeschlossen unter Homomorphismus* $:\Leftrightarrow$ $\mathrm{HOM}(\mathcal{L}) = \mathcal{L}$

abgeschlossen unter Homomorphismus

1.8. Satz:

(i) $\mathcal{L}_3, \mathcal{L}_2, \mathcal{L}_0$ sind abgeschlossen unter Homomorphismus

(ii) $\mathrm{HOM}(\mathcal{L}_1) = \mathcal{L}_0$

(iii) $\mathcal{L}_3, \mathcal{L}_2, \mathcal{L}_1, \mathcal{L}_0$ sind abgeschlossen unter inversem Homomorphismus

Beweis: (i) Sei $G = (V, \Sigma, P, S)$ Grammatik vom Typ $i \in \{0, 2, 4\}$, $h : \Sigma^* \to \Gamma^*$ Homomorphismus. Sei $L = L(G) \in \mathcal{L}_i$.

Beh.: $h(L) \in \mathcal{L}_i$

Beweis: Konstruktion von G' mit $L(G') = L$ und G' ist Grammatik vom Typ i: Erweiterung von h auf $(\Sigma \cup V)^*$:

$$h' : \Sigma \cup V \to \Gamma^* \cup V : x \mapsto \begin{cases} h(x) & \text{falls } x \in \Sigma \\ x & \text{falls } x \in V \end{cases}$$

h' wird auf $(\Sigma \cup V)^*$ erweitert: Für $a_1, \ldots, a_n \in \Sigma \cup V$: $h'(a_1 \ldots a_n) := h'(a_1) \ldots h'(a_n)$.

$G' := (V, \Gamma, P', S)$ mit $P' = \{(h(\alpha), h(\beta)) \,|\, (\alpha, \beta) \in P\}$.

Beh.: $S \overset{*}{\underset{G}{\to}} \alpha \Leftrightarrow S \overset{*}{\underset{G'}{\to}} h(\alpha)$

Beweis: Induktion nach Zahl der Ableitungsschritte r.

IA: $r = 1$: $S \overset{*}{\underset{G}{\to}} \alpha \Leftrightarrow S \underset{G}{\to} \in P \Leftrightarrow S \underset{G'}{\to} h(\alpha) \in P' \Leftrightarrow S \overset{*}{\underset{G'}{\to}} \alpha$

IS: $r \rightsquigarrow r+1$: Die Behauptung gelte für r Ableitungsschritte. Sei $S \overset{*}{\underset{G}{\to}} \alpha_1\alpha_2\alpha_3 \to \alpha_1\beta\alpha_3$ mit $(\alpha_2 \to \beta) \in P$ eine beliebige Ableitung mit $r+1$ Schritten. Nach IV gilt $S \overset{*}{\underset{G'}{\to}} h(\alpha_1\alpha_2\alpha_3) = h(\alpha_1)h(\alpha_2)h(\alpha_3) \underset{G'}{\to} h(\alpha_1)h(\beta)h(\alpha_3) = h(\alpha_1\beta\alpha_3)$, da $h(\alpha_2) \to h(\beta) \in P'$. Analog folgt „$\Leftarrow$“.

Durch Anwendung von h' auf die Produktionen bleiben die Eigenschaften kontextsenitiv bzw. rechtslinear erhalten.

(ii) $\mathrm{HOM}(\mathcal{L}_1) \subseteq \mathcal{L}_0$ (da $\mathcal{L}_0$ abgeschlossen unter HOM), z.z. $\mathcal{L}_0 \subseteq \mathrm{HOM}(\mathcal{L}_1)$. Sei $L \in \mathcal{L}_0$. Konstruktion von $L' \in \mathcal{L}_1$ und Homomorphismus h mit $h(L') = L$. Sei $G = (V, \Sigma, P, S)$ mit $L(G) = L$ und $\forall(\alpha \to \beta) \in P : \alpha \in V^+$ (dies kann leicht durch zusätzliche Variablen erreicht werden).

Sei $G' = (V \cup \{D\}, \Sigma \cup \{d\}, P', S)$ mit

$$\begin{aligned} P' = \quad & \{(u, v) \in P \,|\, |u| \leq |v|\} \\ & \cup \{(u, vD^{|u|-|v|}) \,|\, (u, v) \in P, |u| > |v|\} \\ & \cup \{(AD, DA), (DA, AD) \,|\, A \in V \cup \Sigma\} \cup \{(D, d)\} \end{aligned} .$$

Dann gilt: $S \overset{*}{\underset{G}{\to}} \alpha \Leftrightarrow \exists r \in \mathbb{N} : S \overset{*}{\underset{G'}{\to}} \alpha D^r$.

$\Rightarrow L' = L(G') \subseteq \{d^{r_0} u_1 d^{r_1} u_2 \dots d^{r_{n-1}} u_n d^{r_n} \big| u = u_1 u_2 \dots u_n \in L, r_0, \dots, r_n \in \mathbb{N}_0\}$.

Mit $h(a) := \begin{cases} a & \text{falls } a \in \Sigma \\ \varepsilon & \text{falls } a = d \end{cases}$ gilt $h(L') = L$.

(iii) **Beh.:** $\mathcal{L}_0$, $\mathcal{L}_1$ abgeschlossen gegen h^{-1}

Beweis: Sei $L \in \mathcal{L}_0$ (oder $\in \mathcal{L}_1$). Sei M die TM (oder der LBA), der L akzeptiert. Sei h ein Homomorphismus. Sei M' die Turingmaschine, die aus der Eingabe x zuerst $h(x)$ berechnet und dann M simuliert. Dann akzeptiert M' $h(L)$. Falls M ein LBA, so ist auch M' ein LBA, denn die Anwendung von h verlängert die Eingabe maximal um den konstanten Faktor $c = \max\{|h(a)| \big| a \in \Sigma\}$.

Beh.: $\mathcal{L}_2$, $\mathcal{L}_3$ abgeschlossen gegen h^{-1}

Beweis: Sei M ein endlicher Automat (oder ein 1-PDA). M' liest Schritt für Schritt ein Zeichen a und simuliert M mit der Eingabe $h(a)$.

□

1.9. Satz: Zu jedem $L \in \mathcal{L}_0$ gibt es

(i) $L_1, L_2 \in \mathcal{L}_2$ und Homomorphismus h mit $h(L_1 \cap L_2) = L$

(ii) L_1 mit $\overline{L_1} \in \mathcal{L}_2$ und Homomorphismus h mit $h(L_1) = L$

Beweis: (i) Sei $L \in \mathcal{L}_0$, sei $M = (Q, \Sigma, \Gamma, \delta, q_o, F)$ TM mit $L = L(M)$. Definiere:

$$L_{KF} := \Big\{ \overline{w}\not\in K_1 \not\in \dots \not\in K_r \Big| r \in \mathbb{N} \text{ ungerade}, \forall i \in \{1, \dots, r-1\} : K_i \in \Gamma^* Q \Gamma^*, K_1 = \text{rev}(q_0 w), K_r \in \Gamma^* F \Gamma^*, \forall i \text{ gerade} : K_i \underset{M}{\to} \text{rev}(K_{i+1}), \forall i \text{ ungerade} : \text{rev}(K_i) \underset{M}{\to} K_{i+1} \Big\}$$

Weiter: $\overline{\Sigma} := \{\overline{a} \big| a \in \Sigma\}$ und $h : \Sigma \cup \overline{\Sigma} \cup \Gamma \cup \{\not\in\} \to \Sigma^* : x \mapsto \begin{cases} a & \text{falls } x = \overline{a} \\ \varepsilon & \text{sonst} \end{cases}$.

Dann gilt: $h(L_{KF}) = L(M)$.

Beh.: L_{KF} ist Schnitt zweier kontextfreier Sprachen.

Beweis: Sei L_1 und L_2 folgende Sprachen:

$$L_1 = \{\overline{w}\not\in \text{rev}(q_0 w)\not\in K_2 \not\in \dots \not\in K_{r-1} \not\in K_r \big| K_i \in \Gamma^* Q \Gamma^*, K_r \in \Gamma^* F \Gamma^*, K_{2i} \to \text{rev}(K_{2i+1})\}$$

$$L_2 = \{\overline{w}\not\in K_1 \not\in \dots \not\in K_r \big| K_i \in \Gamma^* Q \Gamma^*, K_r \in \Gamma^* F \Gamma^*, \text{rev}(K_{2i-1}) \to K_{2i}\}$$

Dann gilt: $L = L_1 \cap L_2$ und L_1 und L_2 sind kontextfrei.

(ii) **Beh.:** $\overline{L_{KF}(M)} \in \mathcal{L}_2$

Beweis: $\overline{L_{KF}(M)}$ wird als Vereinigung einer regulären Sprache $\overline{R}$ und einer kontextfreien Sprache L' dargestellt. In $\overline{R}$ sind alle Worte, die ein falsches Format haben, in L' sind Worte, bei denen mindestens ein Übergang falsch ist.
Sei $Y := \Gamma \cup Q \cup \Sigma \cup \overline{\Sigma} \cup \{\text{¢}\}$.

$$R := \{u \in Y \,|\, u = \overline{w}\text{¢}K_1\text{¢}\ldots\text{¢}K_r, r \geq 3, K_i \in \Gamma^* Q \Gamma^*, K_r \in \Gamma^* F \Gamma^*\}$$
$$L' := \{\alpha\text{¢}u_1\text{¢}u_2\text{¢}\beta \,|\, \alpha, \beta \in Y^*, u1, u2 \in \Gamma^* Q \Gamma^*, u_1 \not\xrightarrow[M]{} \mathrm{rev}(u_2)\}$$

Dann gilt $R \in \mathcal{L}_3$ und somit auch $\overline{R} \in \mathcal{L}_3$. Weiter gilt $L' \in \mathcal{L}_2$ (Übungsaufgabe) und $\overline{L_{KF}(M)} = \overline{R} \cup L'$.

□

1.10. Lemma: $\mathcal{L}_2$ ist nicht abgeschlossen gegen Komplementbildung.

Beweis: Annahme: $\mathcal{L}_2$ abgeschlossen gegenüber Komplement. Sei $L \in \mathcal{L}_0$. Dann existiert nach Satz 1.9 L_1 mit $\overline{L_1} \in \mathcal{L}_2$ und Homomorphismus h mit $h(L_1) = L$. Also folgt $L_1 \in \mathcal{L}_2 \Rightarrow L = h(L_1) \in \mathcal{L}_2 \Rightarrow \mathcal{L}_0 \subseteq \mathcal{L}_2$ Widerspruch.

□

1.11. Bemerkung (Abschlußeigenschaften):

	h	h^{-1}	nichtlösch.h	$\cup$	$\cap$	Komplement	Kleene+
$\mathcal{L}_3$	+	+	+	+	+	+	+
$\mathcal{L}_2$	+	+	+	+	−	−	+
$\mathcal{L}_1$	−	+	+	+	+	+*	+
$\mathcal{L}_0$	+	+	+	+	+	−	+

1.12. Definition (Turingmaschinen):

(i) Eine *Turingmaschine* (TM) M ist ein 6-Tupel $(Q, \Sigma, \Gamma, \delta, q_0, F)$ mit — *Turingmaschine*

Q: Endliche Menge von *Zuständen* — *Zustände*

Σ: Endliche Menge, — *Alphabet*

Γ: Endliche Menge, $\Sigma \subset \Gamma$, , $B \in \Gamma \setminus \Sigma$ *Blanksymbol* — *Bandalphabet*

δ: Abbildung *Übergangsrelation*, bei *deterministischen* TMs $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, N, R\}$, bei *nichtdeterministischen* TMs $\delta : Q \times \Gamma \to \mathrm{Pot}(Q \times \Gamma \times \{L, N, R\})$. — *Blanksymbol*, *Übergangsrelation*

$q_0 \in Q$: Startzustand — *deterministisch*

$F \subseteq Q$: Akzeptierende Zustände — *nichtdeterministisch*

*Satz von Immerman, Szelepcsényi, s. Seite 8

linear beschränkter Automat

(ii) Ein *linear beschränkter Automat* (LBA) ist eine TM, die den durch BwB definierten Bandabschnitt bei Eingabe w nicht verläßt.

Endlicher Automat

(iii) Ein *Endlicher Automat* (EA, auch FA*) ist eine Turingmaschine, die die Eingabe nur einmal von links nach rechts lesen kann. $M = (Q, \Sigma, \delta, q_0, F)$ mit $\delta : Q \times \Sigma \to Q$.

Kellerautomat
Kelleralphabet
Kellerendesymbol

(iv) Ein *Kellerautomat* (KA, auch PDA†) ist ein 7-Tupel $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ mit Γ endliches *Kelleralphabet*, $Z_0 \in \Gamma$ *Kellerendesymbol*, $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \subseteq Q \to \mathrm{Pot}(Q \times \Gamma^*)$

Ein Kellerautomat heißt *deterministisch*, falls

(1) $\delta(q, \varepsilon, c) \neq \emptyset \Rightarrow \forall a \in \Sigma : \delta(q, a, c) = \emptyset$ und

(2) $\forall q \in Q, c \in \Gamma, a \in \Sigma \cup \{\varepsilon\} : |\delta(q, a, c)| \leq 1$.

Weitere Kellerautomaten: Ein- bzw. Zwei-Wege (nicht) det. Kellerautomaten mit k Köpfen, Sprachklassen: 1DPDA(k), 2DPDA(k), 1NPDA(k), 2NPDA(k).

Zählmaschine

(v) Eine *Zählmaschine* ist ein Kellerautomat mit nur einem Kellersymbol (außer dem Kellerendesymbol), Sprachklassen: 1DC, 1NC, 2DC, 2NC

1.13. Satz (Zusammenhang Sprachklassen und Automaten): Sei $L \subset \Sigma^*$ eine Sprache. Dann gilt:

$$L \in \mathcal{L}_1 \;\Leftrightarrow\; \exists G \text{ beschränkt mit } L(G) = L \;\Leftrightarrow\; \exists \text{NLBA } M \text{ mit } L = L(M)$$

Beweis: Zu zeigen bleibt die zweite Äquivalenz.

"⇒": Seien $G = (V, \Sigma, P, S)$ beschränkte Grammatik und $S = \varphi_1 \to \cdots \to \varphi_r = w \in \Sigma^*$ eine Ableitung. Da G beschränkt ist, gilt $\forall i \in \{1, \ldots, r-1\} : |\varphi_i| \leq |\varphi_{i+1}|$, insbesondere $\forall i \in \{1, \ldots, r-1\} : |\varphi_i| \leq |w|$, d.h. die Zwischenergebnisse jeder Ableitung in G sind nie länger als das abgeleitete Wort.

Konstruktion eines 2-Band nichtdet. linear beschränkten Automatens M:

(i) Simululiere auf dem zweiten Band eine beliebige Ableitung, ohne dabei die ersten $n = |w|$ Zellen zu verlassen.

(ii) Vergleiche, ob auf beiden Bändern das gleiche Wort steht.

"⇐": Sei $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ ein NLBA mit einem Band. O.B.d.A. ersetze M die beiden Randzeichen a_1 und a_n der Eingabe durch $\overline{a_1}$ bzw. $\overline{a_1}$ und verlasse den so markierten Bereich niemals.

Wir konstruieren eine beschränkte Grammatik $G = (\Sigma, V, P, S)$, deren Variablen teilweise Paare (vergl. 2-Band-TMs) sind. Im ersten Element des Paares bleibt die Eingabe stehen, im zweiten Element wird M simuliert. Dies ergibt die folgenden Produktionen:

*finite automaton
†push down automaton

Startphase: $\forall a \in \Sigma : S \to \binom{a}{q_0 \overline{a}} A,\ A \to \binom{a}{a} A,\ A \to \binom{a}{\overline{a}}$

Simulation: Simulation eines Schrittes von M:

$$\binom{a_1}{b_1}\binom{a_2}{qb_2}\binom{a_3}{b_3} \underset{G}{\to} \begin{cases} \binom{a_1}{b_1}\binom{a_2}{b'}\binom{a_3}{q'b_3} & \text{falls } (q', b', R) \in \delta(q, b_2) \\ \binom{a_1}{b_1}\binom{a_2}{q'b'}\binom{a_3}{b_3} & \text{falls } (q', b', N) \in \delta(q, b_2) \\ \binom{a_1}{q'b_1}\binom{a_2}{b'}\binom{a_3}{b_3} & \text{falls } (q', b', L) \in \delta(q, b_2) \end{cases}$$

Akzeptierphase: Löschen der ersten Spur, falls ein Endzustand erreicht ist: $\forall a \in \Sigma, b \in \Gamma : \binom{a}{b} \to a$ und $\forall q \in F, a \in \Sigma, b \in \Gamma : \binom{a}{qb} \to a$

□

1.14. Bemerkung (Zusammenhang Sprachklassen und Automaten):

(i) Die regulären Sprachen werden von deterministischen und nichtdeterministischen EAs akzeptiert, die kontextfreien Sprachen von den nd PDAs, die deterministisch kontextfreien Sprachen von den det. PDAs, die kontextsensitiven Sprachen von den nd. LBAs (ob auch von den det. LBAs, ist offen), die rekursiv aufzählbaren Sprachen ($\mathcal{L}_0$) von den TMs.

(ii) Es gelten folgende Zusammenhänge:

(1) $\mathcal{L}_3 = 1\text{DFA}(1) = 1\text{NFA}(1) = 2\text{NFA}(1) = 2\text{DFA}(1)$

(2) $\{a^n b^n c^n | n \in \mathbb{N}\} \in 1\text{DFA}(2)$

(3) $\text{HOM}(1\text{DFA}(2)) = \mathcal{L}_0$

(4) $\bigcup_k 2\text{DFA}(k) = \text{DSPACE}(\log n)$, $\bigcup_k 2\text{NFA}(k) = \text{NSPACE}(\log n)$

(5) $1\text{NPDA}(1) = \mathcal{L}_2 = \text{CFL}$, $1\text{DPDA}(1) = \text{DCFL}$

(6) $\bigcup_k 2\text{DPDA}(k) = \bigcup_k 2\text{NPDA}(k) = \text{P}$, $2\text{DPDA}(k) \subseteq \text{TIME}(n^k)$

1.15. Definition:

(i) Eine Funktion $s : \mathbb{N} \to \mathbb{N}$ heißt *bandkonstruierbar*, wenn es eine $s(n)$-platzbeschränkte Turingmaschine gibt, die bei Eingabe w die Binärdarstellung von $s(|w|)$ berechnet. *bandkonstruierbar*

(ii) $\text{DSPACE}(s(n)) := \{L | \exists\ s(n)\text{-bandbeschränkte det. TM } M \text{ mit } L = L(M)\}$, analog $\text{NSPACE}(s(n))$ für nichtdet. TMs

Algorithmus 1 Wortproblem für co(NSPACE($s(n)$))

Eingabe: N: Anzahl erreichbarer Konfigurationen
Ausgabe: w: true falls keine akzeptierende Berechnung gefunden

```
1: z = 0, w =true, K =erste Konfiguration
2: repeat
3:    rate Konfigurationsfolge q_0 w → k_1 → ··· → k_r mit |k_i| ≤ s(n)
4:    if k_r = K then
5:       z = z + 1 {richtig geraten}
6:       if K ∈ Γ*FΓ* then
7:          w =false
8:    K =Nachfolger(K)
9: until nicht w oder K ist keine gültige Konfiguration mehr
10: if z < N then
11:    w =false {nicht alle erreichbaren Konfigurationen richtig geraten}
```

1.16. Satz (Satz von Immerman und Szelepcsényi): Für bandkonstruierbare Funktionen $s : \mathbb{N} \to \mathbb{N}$ gilt: $\text{NSPACE}(s(n)) = \text{co}(\text{NSPACE}(s(n)))$

Speziell: $\mathcal{L}_1 = \text{NSPACE}(n)$ ist abgeschlossen unter Komplement.

Beweis: Sei $s(n)$ bandkonstruierbar, $L \in \text{NSPACE}(s(n))$. Sei M die TM, die L mit Platz $s(n)$ akzeptiert. Dann existiert $c > 0$ mit M akzeptiert jedes $w \in L$ in $2^{cs(|w|)}$ Schritten. Wir konstruieren eine TM M', die $\overline{L}$ akzeptiert, indem sie alle möglichen Berechnungen von M der Länge maximal $2^{cs(|w|)}$ simuliert und akzeptiert, wenn keine Berechnung von M akzepierend war. Eine deterministische TM würde hierzu wahrscheinlich zu viel Platz benötigen, allerdings gelingt die Konstruktion mit Nichtdeterminismus.

Zuerst berechnen wir durch iterative Anwendung von Algorithmus 2 die Anzahl erreichbarer Konfigurationen von M bei Eingabe w und Platzbeschränkung $s(|w|)$ mit t Schritten für $t = 1, 2, \ldots$ bis $N_t = N_{t+1}$ gilt.

Anschließend überprüfen wir, ob keine der erreichbaren Konfigurationen akzeptierend ist (Algorithmus 1). Dazu raten wir zu jeder Konfiguration eine Konfigurationenfolge und benutzen die Anzahl erreichbarer Konfigurationen, um zu überprüfen, ob wir immer richtig geraten haben.

□

1.17. Bemerkung: Es gilt: $\mathcal{L}_0 = \text{HOM}(1\text{DFA}(2))$.

Beweis: Sei $L \in \mathcal{L}_0$ mit TM M und $L = L(M)$.

Algorithmus 2 Berechung von N_{t+1}

Eingabe: t, N_t: Anzahl erreichbarer Konfigurationen mit t Schritten
Ausgabe: w: true falls $y = N_{t+1}$ gilt, y: N_t+1 wenn w=true

```
 1: y = 0, w =true, K =erste Konfiguration
 2: repeat
 3:    z = 0, f =false, L =erste Konfiguration
 4:    repeat
 5:       rate Konfigurationsfolge q_0 w → k_1 → ··· → k_r mit r ≤ t, |k_i| ≤ s(n)
 6:       if k_r = L then
 7:          z = z + 1 {richtig geraten}
 8:          if K = L oder L → K then
 9:             f =true
10:       L =Nachfolger(L)
11:    until f oder L ist keine gültige Konfiguration mehr
12:    if f then
13:       y = y + 1
14:    else
15:       if z < N_t then
16:          w =false
17:    K =Nachfolger(K)
18: until not w or K ist keine gültige Konfiguration mehr
```

$$\tilde{L}_{KF} := \Big\{ \overline{w}\text{¢}K_1\text{¢}\ldots\text{¢}K_r \,\Big|\, r \in \mathbb{N}, K_1 = (q_0 w), K_r \in \Gamma^* F \Gamma^*, \\ \forall i \in \{1,\ldots,r-1\} : K_i \in \Gamma^* Q \Gamma^*, K_i \underset{M}{\rightarrow} K_{i+}1 \Big\}$$

Dann gilt: $\tilde{L}_{KF} \in$ 1DFA(2) und mit h wie im Beweis zu Satz 1.9 folgt durch $L = h(\tilde{L}_{KF})$ die Behauptung. □

2. Kontextfreie Sprachen

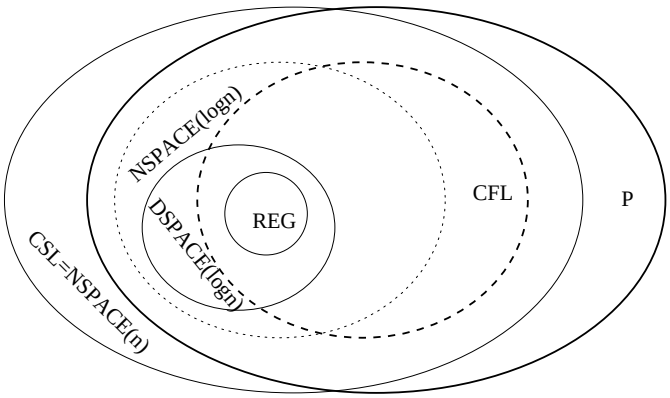
2.1. Definition: Sei $G = (V\Sigma, P, S)$ eine Grammatik.

(i) G heißt *reduziert* $:\Leftrightarrow \forall A \in V : \exists \alpha\beta \in (\Sigma \cup V)^*, \gamma \in \Sigma^* : S \overset{*}{\rightarrow} \alpha A \beta \wedge A \overset{*}{\rightarrow} \gamma$ — *reduziert*

(ii) G ist in *Chomsky-Normalform*, falls für alle Regeln $(A \rightarrow \beta) \in P$ gilt: $A \in V, \exists B, C \in V : \beta = BC \vee \beta \in \Sigma$. — *Chomsky-Normalform*

2.2. Satz: Zu jeder kontextfreien Sprache $L \in$ CFL existiert eine reduzierte Grammatik G in Chomsky-Normalform mit $L(G) = L \setminus \{\varepsilon\}$.

Abbildung 1 Übersicht über die Sprachklassen



Beweis: s. [HU90], Seite 99f. □

2.3. Folgerung: Für jede kontextfreie Grammatik G ist entscheidbar, ob $L(G) = \emptyset$ gilt.

2.4. Satz: Folgende Probleme sind nicht entscheidbar:

(i) G beschränkt, $L(G) = \emptyset$?

(ii) G_1, G_2 kontextfrei, $L(G_1 \cap G_2) = \emptyset$?

(iii) $G = (V, \Sigma, P, S)$ kontextfrei, $L(G) = \Sigma^*$?

(iv) M 2DFA(2), $L(M) = \emptyset$?

Beweis: Beh.: Für eine allgemeine Grammatik G ist $L(G) = \emptyset$ unentscheidbar

Beweis: Reduktion auf das Halteproblem (Übungsaufgabe)

(i) **Annahme:** $L(G) = \emptyset$ entscheidbar. Sei $L \in \mathcal{L}_0$. Nach Satz 1.9 existieren $L_1 \in \mathcal{L}_1$ und h mit $h(L_1) = L$. Sei G die beschränkte Grammatik mit $L(G) = L_1$. Dann gilt: $L = \emptyset \Leftrightarrow L_1 = \emptyset \Leftrightarrow L(G) = \emptyset$. Damit wäre L_0 entscheidbar, dies ist ein Widerspruch.

(ii) **Annahme:** Für G_1, G_2 kontextfrei ist $L(G_1) \cap L(G_2) = \emptyset$ entscheidbar. Sei $L \in \mathcal{L}_0$. Nach Satz 1.9 existieren $L_1, L_2 \in \mathcal{L}_2 : h(L_1 \cap L_2) = L$. Dann gilt: $L = \emptyset \Leftrightarrow h(L_1 \cap L_2) = \emptyset \Leftrightarrow L_1 \cap L_2 = \emptyset$. Damit wäre L_0 entscheidbar, dies ist ein Widerspruch.

□

2.5. Definition (Parikh-Abbildung): Sei $\Sigma = \{a_1, \ldots, a_r\}$ mit $r = |\Sigma|$, sei Anz: $\Sigma \times \Sigma^* \to \mathbb{N}$ die Abbildung, die jedem Buchstaben-Wort-Paar die Anzahl der Vorkommen des Buchstabens in dem Wort zuordnet.

Die *Parikh-Abbildung* ist wie folgt definiert: *Parikh-Abbildung*

$$\Psi : \Sigma^* \to \mathbb{N}_0^{|\Sigma|} : u \mapsto (\mathrm{Anz}(a_1, u), \ldots, \mathrm{Anz}(a_r, u))$$

2.6. Definition: Sei $S \subseteq \mathbb{N}^r$ eine Menge.

S heißt *linear* $:\Leftrightarrow \exists u_0, \ldots, u_p \in \mathbb{N}^p : S = \left\{ u_0 + \sum_{i=1}^{p} x_i u_i \middle| x_i \in \mathbb{N}_0 \right\}$ *linear*

S heißt *semilinear* $:\Leftrightarrow \exists n \in \mathbb{N}, S_1, \ldots, S_n$ linear, $S = \bigcup_{i=1}^{n} S_i$ *semilinear*

2.7. Satz (Satz von Parikh): Sei $L \in \mathrm{CFL}$ eine kontextfreie Sprache.

Dann ist $\Psi(L) := \{\Psi(u) | u \in L\}$ semilinear.

Beweis: Sei $L \in \mathcal{L}_2$ und G eine reduzierte kontextsensitive Grammatik mit $L = L(G)$. G habe keine ε-Regeln. Für alle $U \subseteq V$ sei $L_U := \{v \in L \big| \exists$ Ableitung $S \overset{*}{\to} v$ in der nur Variablen aus U auftreten$\}$. Dann gilt

$$L = \bigcup_{U \subseteq V, S \in U} L_U$$

Wir zeigen im folgenden, dass $\forall U \subseteq V : L_U$ ist semilinear gilt. Im folgenden sei $U \subseteq V$ mit $S \in U$ fest.

Zuerst definieren wir zwei Arten von guten Ableitungen, diese spielen bei der Linearität die Rolle von Start- und Richtungsvektoren.

(i) Eine Ableitung $S \overset{*}{\to} v$ mit $v \in \Sigma^*$ heißt *gut*, falls für den zugehörige Ableitungsbaum gilt:

(1) Wurzel ist mit S markiert, als Markierungen der inneren Knoten treten alle Variablen aus U auf, und

(2) Auf jedem von der Wurzel startenden Weg tritt jede Variable höchstens $r = |U|$-mal auf.

$\Rightarrow$ Ableitungsbäume von guten Ableitungen haben Tiefe maximal r^2. Sei $H = \{v \in \Sigma^* \big| \exists S \overset{*}{\to} v$ gut$\}$. Dann gilt $|H| < \infty$.

(ii) Eine Ableitung $A \overset{*}{\to} uAv$ mit $u, v \in \Sigma^*$, $A \in U$ heißt *gut*, falls gilt:

(1) Im Ableitungsbaum treten nur Variablen aus U auf, und

(2) auf jedem an der Wurzel startenden Weg tritt jede Variable höchstens zweimal auf.

Sei $H_A = \{uv \mid \exists A \xrightarrow{*} uAv \text{ gut mit } uv \neq \varepsilon\}$. Dann gilt $\forall A \in U : |H_A| < \infty$.

Seien $S_1 := \Psi(H) =: \{a_1, \ldots, a_\lambda\}$, $S_2 = \Psi(\bigcup_{A \in U} H_A) =: \{v_1, \ldots, v_r\}$ und

$$S := \bigcup_{i=1}^{\lambda} \left\{ a_i + \sum_{i=1}^{r} x_i v_i \,\middle|\, x_i \in \mathbb{N}_0 \right\}$$

Beh.: $\Psi(L_U) = S$

Beweis: „$\subseteq$“: Sei $s \in S \Rightarrow s = u + \sum_{i=1}^{r} x_i v_i$ mit $u \in S_1$, $v_i \in S_2$, $x_i \in \mathbb{N}_0$. Beweis durch Induktion nach $x = \sum_{i=1}^{r} x_i$.

IA: $x = 0 \Rightarrow \forall i \in \{1, \ldots, r\} : x_i = 0 \Rightarrow s = u \in S_1 = \Psi(H) \subseteq \Psi(L_U)$

IS: Die Aussage gelte für $x \geq 0$. Sei $\sum_{i=1}^{r} x_i = x + 1 \Rightarrow \exists j : x_j > 0$. Sei $s' = u + \sum_{i=1}^{r} y_i$ mit $y_i = \begin{cases} x_i & i \neq j \\ x_i - 1 & i = j \end{cases}$

Dann gilt $x = \sum_{i=1}^{r} y_i$ und somit nach IV $s' \in \Psi(L_U) \Rightarrow \exists w \in L_U : s' = \Psi(w)$.

Es gibt $A \in U : v_j \in \Psi(H_A)$ (nach Definition S_2). Da in L_U nur Wörter sind, für die es eine Ableitung gibt, die alle Variablen aus U benutzt, existiert eine Ableitung für w mit $S \xrightarrow{*} w_1 A w_3$, $A \xrightarrow{*} w_2$ und $w = w_1 w_2 w_3$. Weiter existiert $w' \in H_A$ mit $v_j = \Psi(w')$, d.h. $A \xrightarrow{*} w_1' A w_2'$ mit $w' = w_1' w_2'$.

Zusammen folgt $S \xrightarrow{*} w_1 A w_3 \xrightarrow{*} w_1 w_1' A w_2' w_3 \xrightarrow{*} w_1 w_1' w_2 w_2' w_3 \in L_U$ und $\Psi(w_1 w_1' w_2 w_2' w_3) = \Psi(w_1 w_2 w_3) + \Psi(w_1' w_2') = \Psi(w) + \Psi(w') = s' + v_j = s \Rightarrow s \in \Psi(L_U)$.

„$\supseteq$“: Sei $w \in L_U$. Beweis von $\Psi(w) \in S$ durch Induktion nach $n = |w|$:

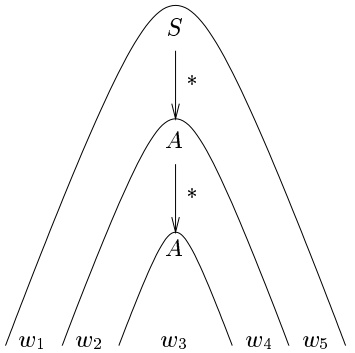
IA: $n = |w| = 1 \Rightarrow w \in H \Rightarrow \Psi(w) \in S_1 \subseteq S$

IS: $n \rightsquigarrow n + 1$: $|w| = n + 1$, $\exists$ Ableitung $S \xrightarrow{*} w$ in der genau Variablen aus U auftreten.

Falls diese Ableitung gut ist, gilt $w \in H \Rightarrow \Psi(w) \in S_1 \subseteq S$.

Sei diese Ableitung nicht gut, d.h. $\exists$ Variable A, die auf einem Weg des Ableitungsbaumes mindestens $(r + 1)$-mal auftritt.

Idee: Wir suchen nun eine Ableitung der folgenden Form:



(i) in $S \overset{*}{\to} w_1 A w_5 \overset{*}{\to} w_1 w_3 w_5$ treten alle Variablen aus U auf

(ii) $w = w_1 w_2 w_3 w_4 w_5$, $|w_2 w_4| > 0$

(iii) $A \overset{*}{\to} w_2 A w_4$ ist gut

Dann gilt: $\Psi(w) = \Psi(w_1 w_3 w_5) + \Psi(w_2 w_4)$ mit $\Psi(w_2 w_4) \in S_2$, $\Psi(w_1 w_3 w_5) \in S$ nach IV

Seien $\alpha_0, \ldots, \alpha_r$ Knoten mit Variablen A auf einem Weg von der Wurzel zu einem Blatt. Sei τ_i der maximale Teilbaum des Ableitungsbaumes, der α_i enthält, aber nicht die Kinder von α_i. Seien K_i die Menge aller Variablen, die in Baum τ_i auftreten.
Dann gilt: $\forall i \in \{0, \ldots, r-1\} : K_i \subseteq K_{i+1} \subseteq U$ und $|K_0| \geq 1$
$\Rightarrow \exists j : K_j = K_{j+1}$
$\Rightarrow$ Mit den Bezeichnungen der Skizze (das obere A steht an Knoten α_j, das untere A an Knoten α_{j+1}) gilt: In der Ableitung $S \overset{*}{\to} w_1 A w_5 \overset{*}{\to} w_1 w_3 w_5$ treten genau die Variablen aus S auf, d.h. $w_1 w_3 w_5 \in L_U$ und damit $\Psi(w_1 w_3 w_5) \in S$ nach IV.
Falls $A \overset{*}{\to} w_2 A w_4$ gut ist, gilt $w_2 w_4 \in H_A$ und damit $\exists l : \Psi(w_2 w_4) = v_l \Rightarrow \Psi(w) = \Psi(w_1 w_3 w_5) + v_j \in S$ und es folgt die Behauptung für $n+1$.
Sei $A \overset{*}{\to} w_2 A w_4$ nicht gut $\Rightarrow \exists$ Variable B, die auf einem Weg mindestens dreimal auftritt.
Betrachte nun den Ableitungsbaum mit den Knoten β_i, an denen die B stehen und iteriere diese Analyse mit β_i statt α_i. Da die untersuchten Teilbäume bei jeder Iteration echt kleiner werden, existiert eine Zerlegung mit einer guten Ableitung.

□

2.8. Bemerkung: Die Umkehrung des Satzes von Parikh gilt i.a. nicht.

Beweis: $L = \{a^n b^n c^n\}$ ist nicht kontextfrei (Beweis z.B. mit Pumping-Lemma). Aber $\Psi(L) = \{(n,n,n) \big| n \in \mathbb{N}\} = \{(1,1,1) + x(1,1,1) \big| x \in \mathbb{N}_0\}$ ist linear und damit auch semilinear. □

2.9. Folgerung: Sei $L \in \mathcal{L}_2$ mit $L \subseteq \{a\}^*$. Dann gilt $L \in \mathcal{L}_3$.

Beweis: Sei $L \subseteq \{a\}^*$. Dann gilt $\Psi(L) \sim L$. Sei $\Psi(L) = \{u_0 + \sum_{i+1}^{r} x_i u_i \big| x_i \in \mathbb{N}_0\}$. Die Frage „Gibt es für gegebenes $z \in \mathbb{N}$ x_i mit $z = u_0 + \sum_{i+1}^{r} x_i u_i$?“ kann von einem endlichen Automaten entschieden werden ($u_0, \ldots, u_r$ werden im endlichen Zustand gespeichert). □

2.10. Satz: Sei $L \subseteq \Sigma^*$. Dann gilt:

(i) $\exists G$ kontextfreie Grammatik mit $L = L(G) \Leftrightarrow \exists$ NPDA M mit $L(M) = L$
$\Leftrightarrow \exists$ NPDA M mit $N^*(M) = L$

(ii) Zu jedem det. KA M gibt es einen det. KA M' mit $N(M) = L(M')$

Beweis: (i) (1) Sei $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ gegeben. Konstruktion von einem PDA $M' = (Q', \Sigma, \Gamma', \delta', q_0', Z_0', \{t\})$ mit $L(M') = N(M)$. Sei $Q' = Q \cup \{t, q_0'\}$, $\Gamma' = \Gamma \cup \{Z_0'\}$

$$\delta'(q,a,Z) = \begin{cases} \delta(q,a,Z) & \text{falls } Z \neq Z_0 \\ & \text{oder } \delta(q,a,Z) \cap Q \times \varepsilon = \emptyset \\ \delta(q,a,Z) \cup (t, Z_0) & \text{sonst} \end{cases}$$

(2) Sei $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ gegeben. Konstruktion von einem PDA $M' = (Q', \Sigma, \Gamma', \delta', q_0', Z_0, \emptyset)$ mit $N(M') = L(M)$. Immer wenn M in einen akzeptierenden Zustand gelangt, rät M', ob er den Keller leert. $Q' = Q \cup \{t\}$,

$$\delta'(q,a,Z) = \begin{cases} \delta(q,a,Z) & \text{falls } \delta(q,a,Z) \cap F \times \Gamma^* = \emptyset \\ \delta(q,a,Z) \cup \{(t,Z)\} & \text{sonst} \end{cases}$$

$\forall Z \in \Gamma : \delta'(t, \varepsilon, Z) = (t, \varepsilon)$

(3) Sei M gegeben. Konstruktion einer kontextfreien Grammatik G mit $L(G) = N(M)$.
Idee: Um mit leerem Keller zu akzeptieren, ist wichtig, welcher Zustand p erreicht wird und welche Eingabe x gelesen wird, um ein bestimmtes Symbol vom Keller zu löschen. Es soll folgende Äquivalenz gelten:

$$(q, Z, p) \overset{*}{\underset{G}{\to}} x \Leftrightarrow (q, x, Z) \overset{*}{\underset{M}{\to}} (p, \varepsilon, \varepsilon) \qquad (1)$$

Konstruktion der Produktionen von G:

*Akzeptieren mit leerem Keller

(a) $\forall q \in F : S \to (q_0, Z_0, q)$

(b) $\forall q \in Q, a \in \Sigma \cup \{\varepsilon\}, Z \in \Gamma : \forall (q', \varepsilon) \in \delta(q, a, Z) : (q, Z, q') \to a$

(c) $\forall q \in Q, a \in \Sigma \cup \{\varepsilon\}, Z \in \Gamma : \forall (q', B_1 \dots B_r) \in \delta(q, a, Z) : \forall q_1, \dots, q_r \in Q : q_r(q, Z, q_r) \to a(q', B_1, q_1)(q_1, B_2, q_2) \dots (q_{r-1}, B_r, q_r)$

Beh.: Diese Produktionen implizieren die Äquivalenz (1).

Beweis: „$\Leftarrow$“: Es gelte $(q, x, Z) \overset{k}{\underset{M}{\to}} (p, \varepsilon, \varepsilon)$. Beweis von $(q, Z, p) \overset{*}{\underset{G}{\to}} x$ mit Induktion nach k:

IA: $k = 1 \Rightarrow x \in \Sigma \cup \{\varepsilon\} \Rightarrow (p, \varepsilon) \in \delta(q, x, Z) \Rightarrow (q, Z, p) \underset{G}{\to} x$.

IS: $k - 1 \rightsquigarrow k$, $k > 1$: Betrachte den ersten Berechnungsschritt von M: $(q_1, B_1 \dots B_r) \in \delta(q, a, Z)$ mit $a \in \Sigma \cup \{\varepsilon\}$ und $\exists x_1, \dots, x_r$ und $q_i \in Q : x = ax_1 \dots x_r, \forall i \in \{1, \dots, r+1\} : (q_i, x_i, B_i) \overset{*}{\underset{M}{\to}} (q_{i+1}, \varepsilon)$ mit $q_{r+1} = p$, da M die im ersten Schritt auf den Keller geschriebenen Symbole irgendwann löschen muss.
Nach IV gilt $(q_i, B_i, q_{i+1}) \overset{*}{\underset{G}{\to}} x_i$

$\Rightarrow (q, Z, p) \underset{G}{\to} a(q_1, B_1, q_2)(q_2, B_2, q_3) \dots (q_r, B_r, p) \overset{*}{\underset{G}{\to}} ax_1 \dots x_r = x$.

(4) Sei G kontextfreie Grammatik. Beweis der Existenz eines PDAs mit $L(M) = L(G)$ s. [HU90], Seite 123f.

(ii) analog dem nichtdeterministischen Fall mit

$$\delta'(q, a, Z) = \begin{cases} \delta(q, a, Z) & \text{falls } Z \neq Z_0 \text{oder } \delta(q, a, Z) \cap Q \times \varepsilon = \emptyset \\ \delta(t, Z_0) & \text{sonst} \end{cases}$$

□

2.11. Definition:

(i) Eine Grammatik heißt *eindeutig*, falls es für jedes $w \in L(G)$ genau eine Linksableitung gibt. Sonst heißt G *mehrdeutig*. *eindeutig* *mehrdeutig*

(ii) Eine Sprache L heißt *eindeutig*, falls es eine eindeutige Grammatik G mit $L(G) = L$ gibt, sonst heißt L *mehrdeutig*.

2.12. Satz:

(i) DCFL ist abgeschlossen unter Komplementbildung.

(ii) DCFL ist abgeschlossen unter inversen Homomorphismen und unter Schnittbildung mit regulären Sprachen.

(iii) DCFL ist nicht abgeschlossen unter Homomorphismus, Vereinigung, Schnitt, Konkatenation und Kleenschem Abschluss.

Beweis: (i) (vergl. [HU90], Seite 257ff) Im Gegensatz zu endlichen Automaten kann man bei Kellerautomaten nicht einfach die Menge der akzeptierenden Zustände ändern, um das Komplement einer Sprache zu akzeptieren, da es mehrere Gründe gibt, warum ein Kellerautomat ein Wort nicht akzeptiert. Ein DKA M akzeptiert ein Wort w nicht, wenn einer der folgenden Fälle auftritt:

(1) M bewegt sich niemals über einen Punkt der Eingabe hinweg, weil ein Übergang nicht definiert ist oder M eine unendliche ε-Bewegung macht.

(2) M macht am Ende des Wortes eine unendliche ε-Bewegung, in der kein akzeptierender Zustand erreicht wird.

Zuerst werden wir einen DKA M konstruieren, der immer die gesamte Eingabe liest und keine endlosen ε-Bewegungen macht. Anschließend modifizieren wir M so, daß die Wahl zwischen ε-Bewegung und Lesen eines Zeichens nur vom Zustand und nicht mehr vom Kellerinhalt abhängt.

Definition: M ist in *Normalform*, falls $\forall a \in \Sigma \cup \{\varepsilon\}, q \in Q, Z \in \Gamma$ mit $\delta(q, a, Z) = (q', \lambda)$ gilt:

(1) $|\lambda| \leq 2$

(2) $|\lambda| = 1 \Rightarrow \lambda = Z$

(Dies bedeutet, daß M entweder ein Kellersymbol löscht (pop), den Keller unverändert lässt, oder ein neues Zeichen auf den Keller schreibt (push).)

Beh.: $\forall$DKA $M : \exists$DKA M' in Normalform mit $L(M) = L(M')$

Beweis: Sei $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ ein DKA. Sei $M' = (Q', \Sigma, \Gamma, \delta', q_0, Z_0, F)$ mit $Q' = Q \cup \{\text{neue Zustände durch Definition der Regeln, s.u.}\}$.
M' simuliert M mit folgenden Regeln:

(1) $\forall q \in Q, a \in \Sigma \cup \{\varepsilon\}, Z \in \Gamma : \delta(q, a, Z) = (q', A_1 \dots A_n)$ wird simuliert durch $\delta'(q, a, Z) = (q^{n-1}, A_{n-1}A_n), \dots, \delta'(q^2, \varepsilon, A_2) = (q', A_1A_2)$ mit neuen Zuständen $q^2, \dots, q^{n-1}$.

(2) $\forall q \in Q, a \in \Sigma \cup \{\varepsilon\}, Z \in \Gamma : \delta(q, a, Z) = (q', A), A \neq Z$ wird simuliert durch $\delta'(q, a, Z) = (\hat{q}, BA)$ und $\delta'(\hat{q}, \varepsilon, B) = (q', \varepsilon)$ mit neuem Zustand $\hat{q}$ und beliebigem $B \in V$.

Lemma: Sei M ein DKA. Dann existiert ein DKA M' mit $L(M) = L(M')$ und M' liest jede Eingabe komplett.

Beweis: Sei M ein DKA in Normalform. Sei $M' = (Q', \Sigma, \Gamma, \delta', q_0', Z_0', F')$ mit $Q' = Q \cup \{q_0', d, f\}$, $\Gamma' = \Gamma \cup \{Z_0'\}$, $F' = F \cup \{f\}$ mit folgenden Regeln:

(1) $\delta'(q_0', \varepsilon, Z_0') = (q_0, Z_0X_0)$ (damit Löschen des alten Kellerendesymbols erkannt werden kann)

(2) $\forall q \in Q, a \in \Sigma, Z \in \Gamma$ mit $\delta(q, a, Z)$ und $\delta(q, \varepsilon, Z)$ nicht definiert: $\delta'(q, a, Z) = (d, Z)$ (dies verhindert Abbrüche durch nicht definierte Übergänge, d steht für dead und kennzeichnet, das M' nicht akzeptieren darf).

(3) $\forall q \in Q, a \in \Sigma : \delta'(q, a, X_0) = (d, X_0)$ (verhindert Abbruch durch leeren Keller)

(4) $\forall a \in \Sigma, Z \in \Gamma' : \delta'(d, a, Z) = (d, Z)$ (im Zustand d das Wort bis zum Ende lesen, Keller nicht verändern)

(5) $\forall q \in Q, Z \in \Gamma$ mit (q, ε, Z) initiiert eine unendliche Rechnung von M: $\delta'(q, \varepsilon, Z) = (q', Z)$ mit $q' = d$ falls in der Rechnung kein Endzustand auftritt und $q' = f$ sonst. Für die Existenz von M' wird nicht benötigt, ob man die Frage nach dem Endzustand entscheiden kann, da man nur endlich oft diese Frage entscheiden muss, und für jede mögliche Antwort existiert ein DKA, einer dieser DKAs ist M'.

(6) $\forall a \in \Sigma, Z \in \Gamma' : \delta'(f, a, Z) = (d, Z)$ (f bedeutet, endlose ε-Bewegung mit akzeptierenden Zustand, d.h. M würde kein weiteres Zeichen lesen und niemals das Wort akzeptieren, falls noch ein Zeichen kommt).

(7) $\forall q \in Q, a \in \Sigma, Z \in \Gamma$ mit $\delta'(q, a, Z)$ noch nicht definiert: $\delta'(q, a, Z) = \delta(q, a, Z)$

Lemma: Sei M ein DKA. Dann existiert ein DKA $M' = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ mit $L(M) = L(M)$ und

(1) M' liest jede Eingabe bis zum Ende

(2) $\forall q \in Q, a \in \Sigma \cup \{\varepsilon\}, X \in \Gamma : \delta(q, a, X) = (p, \gamma)$
$\Rightarrow \gamma \in \{\varepsilon, X, ZX \,|\, Z \in \Gamma\}$ (Kelleränderungen nur push, pop)

(3) $\forall q \in Q, X \in \Gamma : \delta(q, \varepsilon, X) \neq \emptyset \Rightarrow \forall Z \in \Gamma \delta(q, \varepsilon, Z) \neq \emptyset$
(ε-Bewegungen hängen nur vom Zustand ab)

Beweis: Z.z. bleibt (3). Dazu wird das oberste Kellersymbol im Zustand gespeichert. Sei $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ ein DKA. Konstruktion von $M' = (Q', \Sigma, \Gamma', \delta', q_0', Z_0', F')$. Sei $Q' = Q \times \Gamma$, $q_0' = [q_0, Z_0]$, $\Gamma' = \Gamma \cup \{Z_0'\}$, $F' = F \times \Gamma$. Folgende Äquivalenz soll gelten:

$$(q, w, Z_0) \overset{*}{\underset{M}{\to}} (p, v, X\varphi) \Leftrightarrow ([q_0, Z_0], w, Z_0') \overset{*}{\underset{M'}{\to}} ([p, X], v, \varphi Z_0)$$

Dazu definieren wir folgende Regeln:

(1) $\forall q \in Q, a \in \Sigma, X \in \Gamma$ mit $\delta(q, a, X) = (p, \varepsilon)$ definiere $\forall Y \in \Gamma$: $\delta'([q, x], a, Y) = ([p, Y], \varepsilon)$

(2) $\forall q \in Q, a \in \Sigma, X \in \Gamma$ mit $\delta(q, a, X) = (p, Z)$ definiere $\forall Y \in \Gamma'$: $\delta'([q, X], a, Y) = ([p, Z], Y)$

(3) $\forall q \in Q, a \in \Sigma, X \in \Gamma$ mit $\delta(q, a, X) = (p, Z_1 Z_2)$ definiere $\forall Y \in \Gamma$: $\delta'([q, X], a, Y) = ([p, Z_1], Z_2 Y)$

Nun kommen wir zum Beweis der Abgeschlossenheit der DCFL.

Sei $L \in$ DCFL und $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ ein DKA, der die Eigenschaften des letzten Lemmas erfülle und sei $L(M) = L$. Wir konstruieren nun einen DKA $M' = (Q', \Sigma, \Gamma, \delta, q_0', Z_0, F')$, der $\overline{L}$ akzeptiert. Dazu definieren wir: $Q' := Q \times \{1,2,3\}$, $F' = Q \times \{3\}$, $q_0 := [q_0, 1]$ falls $q_0 \in F$ und $q_0 := [q_0, 2]$ sonst. M' befindet sich immer in einem Zustand $[q, 2]$, falls M das Wort nicht akzeptieren würde. Falls M einen akzeptierenden Zustand erreicht, merkt sich M' dies mit $[q, 1]$ und bleibt in $[Q, 1]$, bis das nächste Zeichen gelesen wird. Falls in $[q, 2]$ ein Zeichen gelesen wird, geht M' zuerst mit einer ε-Bewegung in $[q, 3]$, damit M' akzeptiert, falls die Eingabe zu Ende ist. Dies führt zu folgenden Regeln:

(1) $\forall q \in Q, Z \in \Gamma$ mit $\delta(q, \varepsilon, Z) = (p, \gamma)$ definiere $\forall k \in \{1,2\}$:
$\delta'([q,k], \varepsilon, Z) = ([p, k'], \Gamma)$ mit $k' = \begin{cases} 1 & \text{falls } k = 1 \text{ oder } p \in F \\ 2 & \text{sonst} \end{cases}$

(2) $\forall q \in Q, a \in \Sigma, Z \in \Gamma$ mit $\delta(q, a, Z) = (p, \gamma)$ definiere
$\delta'([q,2], \varepsilon, Z) = ([q,3], Z)$ (akzeptieren falls Wortende) und
$\delta'([q,1], a, Z) = \delta'([q,3], a, Z) = ([p,k], \gamma)$ mit $k = \begin{cases} 1 & \text{falls } p \in F \\ 2 & \text{sonst} \end{cases}$

$\Rightarrow M'$ akzeptiert $\overline{M}$.

(ii) **Beh.:** DCFL ist abgeschlossen unter inversen Homomorphismen

Beweis: Seien $L \in$ DCFL, M ein DKA mit $L = L(M)$ und h ein Homomorphismus. $h^{-1}(L) = \{w \mid h(w) \in L\}$. M' simuliert bei Eingabe von w den DKA M mit Eingabe $h(w)$, indem bei jedem gelesenen Zeichen a M auf $h(a)$ (im Zustand gespeichert) simuliert wird.

Beh.: Sei $L \in$ DCFL, R regulär. Dann gilt $L \cap R \in$ DCFL.

Beweis: Seien DKA $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, DFA $M' = (Q', \Sigma, \delta', q_0', F')$ mit $L(M) = L$ und $L(M') = R$ gegeben.
Konstruktion von $\hat{M} = (\hat{Q}, \Sigma, \Gamma, \hat{\delta}, \hat{q}_0, Z_0, \hat{F})$: $\hat{Q} = Q \times Q'$, $\hat{q}_0 = [q_0, q_0']$, $\hat{F} = F \times F'$ und Übergängen

(1) $\forall q \in Q, q' \in Q', a \in \Sigma, Z \in \Gamma$ mit $|\delta(q, a, Z)| = 1$:
$\hat{\delta}([q, q'], a, Z) = ([\delta_1(q, a, Z), \delta'(q', a)], \delta_2(q, a, Z))$

(2) $\forall q \in Q, q' \in Q', Z \in \Gamma$ mit $|\delta(q, \varepsilon, Z)| = 1$:
$\hat{\delta}([q, q'], \varepsilon, Z) = ([\delta_1(q, \varepsilon, Z), q'], \delta_2(q, \varepsilon, Z))$

(dabei beschreibt $\delta_i(q, a, Z)$ die i-te Komponente von $\delta(q, a, Z)$).

(iii) **Schnittbildung:** $L_1 := \{a^n b^n c^m \mid n, m \in \mathbb{N}\}$, $L_2 := \{a^n b^m c^m \mid n, m \in \mathbb{N}\}$. Dann gilt: $L_1, L_2 \in$ DCFL und $L_1 \cap L_2 = \{a^n b^n c^n \mid n \in \mathbb{N}\} \notin \mathcal{L}_2$.

Vereinigung: $L_1 = \{a^n b^n c^m \mid n, m \in \mathbb{N}\}$, $L_2 = \{a^n b^m c^m \mid n, m \in \mathbb{N}\}$, dann gilt $L_1 \cup L_2 \notin$ DCFL (Übungsaufgabe)

Konkatenation: Seien L_1, L_2 wie bei Vereinigung definiert. Dann gilt $dL_1 \circ L_2 \in$ DCFL. Sei $L_3 := \{a\}^* \circ (dL_1 \circ L_2 \in$ DCFL). Annahme: $L_2 \in$ DCFL $\Rightarrow L_2 \cap \{d\}a^*b^*c^* \in$ DCFL $\Rightarrow dL_1 \cup dL_2 \in$ DCFL Widerspruch (vergl. Vereinigung)

Homomorphismen: Seien L_1, L_2 wie bei Vereinigung definiert. Sei $L_3 := dL_1 \circ L_2 \in$ DCFL, $h(d) = \varepsilon$, $h(a) = a$, $h(b) = b$, $h(c) = c$. Dann ist $h(L_3) = L_1 \cup L_2 \notin$ DCFL.

Kleenscher Abschluss: Seien L_1, L_2 wie bei Vereinigung definiert. Sei $L_3 := \{a\} \cup aL_2 \cup L_2 \in$ DCFL. Dann gilt: $L_3 \notin$ DCFL (Übungsaufgabe)

□

2.13. Satz: Sei $G = (V, \Sigma, P, S)$ eine beschränkte Grammatik. Dann ist das Wortproblem für G in Zeit $O(n|V \cup \Sigma|^n)$ lösbar.

Beweis: Sei $w \in \Sigma^*$. Definiere Graphen $H = (U, E)$ mit $U = (V \cup \Sigma)^{\leq n}$ und $E = \{(\alpha, \beta) | \alpha, \beta \in U, \alpha \underset{G}{\rightarrow} \beta\}$. Dann gilt $w \in L(G) \Leftrightarrow \exists$ Weg von S nach w in H. Aufwand für die Wegsuche $O(|E|) = O(n|U| \cdot |P|) = O(n|V \cup \Sigma|^n)$. □

3. Syntaxanalyse kontextfreier Grammatiken

3.1. Satz: Der Early-Algorithmus arbeitet korrekt und hat die Laufzeit $O(n^3)$.

Beweis: z.z.: $\forall 0 \leq j \leq n$:

$$[A \to \alpha.\beta, i] \in I_j \Leftrightarrow \exists \mu, \delta : S \overset{*}{\to} \mu A\delta, \mu \overset{*}{\to} a_1 \dots a_i, \alpha \overset{*}{\to} a_{i+1} \dots a_j, A \to \alpha\beta$$

„⇒“: Beweis durch Induktion nach j und der Zahl r, die angibt, als wievieltes Element es berechnet wurde.

IA ($j = 0$): Regeln nach R1 haben die Form $[S \to .\alpha, 0] \in I_0$, dann gilt $S \overset{*}{\to} \varepsilon S$ und $\varepsilon \overset{*}{\to} \varepsilon = a_1 \dots a_0$.

IS ($j = 0$, $r \rightsquigarrow r + 1$): Betrachte zwei Fälle:

(i) Element wurde durch R3 aufgenommen: $[B \to \alpha A., 0] \in I_0$ wird erzeugt aus $[A \to \gamma., 0] \in I_0$, $[B \to \alpha.A\beta, 0] \in I_0$, dann gilt nach IV $S \overset{*}{\to} \mu B\gamma, \mu \overset{*}{\to} \varepsilon\gamma \overset{*}{\to} \varepsilon, A \to \gamma \Rightarrow S \overset{*}{\to} \mu B\delta, \mu \overset{*}{\to} \varepsilon, \alpha A \overset{*}{\to} \varepsilon$

(ii) Element wurde durch R4 aufgenommen: $[B \to .\gamma, 0] \in I_0$ wird erzeugt aus $[A \to \alpha.B\beta, 0] \in I_0, B \to \gamma$, nach IV: $S \overset{*}{\to} \mu A\delta, \mu \overset{*}{\to} \varepsilon, \alpha \to \varepsilon, A \to \alpha B\beta \Rightarrow S \overset{*}{\to} \mu\alpha B\beta\delta, \mu\alpha \overset{*}{\to} \varepsilon$

Algorithmus 3 Early-Algorithmus (ca. 1970)

Eingabe: $w = a_1 a_2 \dots a_n$, $n = |w|$

Ausgabe: $0 \leq j \leq n : [A \to \alpha.\beta, i] \in I_j \Leftrightarrow \exists \mu, \delta : S \overset{*}{\to} \mu A \delta \wedge \mu \overset{*}{\to} a_1 \dots a_i \wedge \alpha \overset{*}{\to} a_{i+1} \dots a_j \wedge A \to \alpha\beta$

```
 1: ∀j : I_j = ∅
 2: for j = 0 to n do
 3:   if j = 0 then {R1}
 4:     for all (S → α) ∈ P do
 5:       I_0 = I_0 ∪ [S → .α, 0]; push([S → .α, 0])
 6:   else {R2}
 7:     for all [B → α.aβ, i] ∈ I_{j-1}, a = a_j do
 8:       I_j = I_j ∪ {[B → αa.β, i] ∈ I_{j-1}}
 9:       push [B → αa.β, i]
10:   while not empty do
11:     x =pop
12:     if x = [A → γ., i] then {R3}
13:       for all [B → α.Aβ, k] ∈ I_i do
14:         I_j = I_j ∪ {[B → αA.β, k]}, push, falls neues Element
15:     if x = [A → α.Bβ, i] then {R4}
16:       for all (B → γ) ∈ P do
17:         I_j = I_j ∪ {[B → .γ, j]}, push, falls neues Element
```

IS: Die Behauptung gelte für alle $j' < j$ und $j' = j, r' < r$.

(i) Erzeugung des r-ten Elements von I_j durch R2: $[B \to \alpha a_j.\beta, i] \in I_j$ wird erzeugt aus $[B \to \alpha.a_j\beta, i] \in I_{j-1}$, dann gilt nach IV:
$S \overset{*}{\to} \mu B \delta, \mu \overset{*}{\to} a_1 \dots a_i, \alpha \overset{*}{\to} a_{i+1} \dots a_{j-1}$
$\Rightarrow S \overset{*}{\to} \mu B \delta, \mu \overset{*}{\to} a_1 \dots a_i, \alpha a \overset{*}{\to} a_{i+1} \dots a_j$

(ii) Erzeugung des r-ten Elements von I_j durch R3: $[B \to \alpha A.\beta, k] \to I_j$ wird erzeugt aus $[A \to \gamma., i] \in I_j, [B \to \alpha.A\beta, k] \in I_i$, dann gilt nach IV:
$S \overset{*}{\to} \mu B \delta, \mu \overset{*}{\to} a_1 \dots a_k, \alpha \overset{*}{\to} a_{k+1} \dots a_i, A \to \gamma \overset{*}{\to} a_{i+1} \dots a_j, B \to \alpha A \beta$
$\Rightarrow S \overset{*}{\to} \mu \alpha A \beta \delta, \alpha A \overset{*}{\to} a_{k+1} \dots a_j$

(iii) Erzeugung des r-ten Elements von I_j durch R4: $[B \to .\gamma, j] \in I_j$ wird erzeugt aus $[A \to \alpha.B\beta, i] \in I_j, B \to \gamma$, dann gilt nach IV:
$S \overset{*}{\to} \mu A \delta, \mu \overset{*}{\to} a_1 \dots a_i, \alpha \overset{*}{\to} a_{i+1} \dots a_j, A \to \alpha B \beta$
$\Rightarrow S \overset{*}{\to} \mu A \delta \to \mu \alpha A \beta \delta, \mu\alpha \overset{*}{\to} a_1 \dots a_j$

„$\Leftarrow$“: siehe [MAK], Seite 72ff.

Laufzeit: $n + 1$ Schleifendurchläufe mit Kosten jeweils $O(n)$ für R1, R2, R4 und $O(n^2)$ für und R3, zusammen also $O(n^3)$, falls die Schleife $\forall [B \to \alpha A.\beta, k] \in I_j$

effizient ausgeführt werden kann, z.B. mit Mengen $D[j,j] = \{[B \to \alpha A.\beta, k] \in I_j\}$. □

3.2. Lemma:

(i) Zu jedem $L \in \text{PSPACE}$ gibt es ein $L' \in \text{DSPACE}(n)$ mit $L \leq L'$.

(ii) Falls $\text{DSPACE}(n) \subseteq P \Rightarrow \text{PSPACE} \subseteq P$.

Beweis: (i) Sei $L \in \text{PSPACE} \Rightarrow \exists p \in \mathbb{N} : L \in \text{DSPACE}(n^p)$. O.B.d.A. $1 \notin \Sigma$. Dann gilt für

$$L' := \{w\alpha \big| w \in L, \alpha \in \{1\}^* \text{ mit } |w\alpha| = |w|^p\}$$

Sei M die TM, die L mit Platzbeschränkung n^p akzeptiert. Sei M' die TM, die zuerst die Eingabe x auf die Form $x = w\alpha$ mit $w \in \Sigma^*$, $\alpha \in \{1\}^*$ und $|x| = |w|^p$ überprüft und anschließend M auf w simuliert. Dann akzeptiert M' die Sprache L' mit Platzbedarf $O(|w|)$.

(ii) Folgt direkt aus (i) und der Definition von $L \leq L'$.

□

3.3. Satz: Das Wortproblem für LIN ist in Zeit $O(n^2)$ lösbar.

Beweis: Sei $L \in \text{LIN}$, sei $G = (\Sigma, V, P, S)$ lineare Grammatik mit $L = L(G)$. O.B.d.A. enthalte G nur Regeln der Form $A \to aB$, $a \to Ba$, $A \to a$ mit $a \in \Sigma, A, B \in V$. Sei $w = a_1 \dots a_n \in \Sigma^*$.

Idee: Berechnung von Mengen $E[i,j] = \{A \big| A \overset{*}{\to} a_i \dots a_j\}$ mittels dyn. Programmieren mit $r = j - i = 0, 1, \dots$.

for $i = 1$ **to** n **do**
  $E[i,i] = \{A \big| A \underset{G}{\to} a_i\}$
for $r = 1$ **to** $n-1$ **do**
  for $i = 1$ **to** $n - r$ **do**
    $E[i, i+r] = \{A \to a_i B \big| B \in E[i+1, i+r]\} \cup \{A \to Ba_{i+r} \big| B \in E[i, i+r-1]\}$

Dann gilt: $w \in L \Leftrightarrow S \in E[1,n]$ □

3.4. Satz (Satz von Savitch): Sei $s : \mathbb{N} \to \mathbb{N}$ bandkonstruierbar mit $s(n) \geq \log(n)$. Dann gilt:

$$\text{NSPACE}(s(n)) \subseteq \text{DSPACE}(s(n)^2)$$

Beweis: Sei $L \in \text{NSPACE}(s(n))$. Sei $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ eine nichtdet. TM, die L mit Platz $s(n)$ akzeptiert. O.B.d.A. sei $F = \{f\}$.

Sei $w \in \Sigma^*$. Sei $K = Q \times \{1, \ldots, n\} \times \Gamma^{s(n)} \times \{1, \ldots, s(n)\}$.

Wenn M w mit Platz $s(n)$ akzeptiert, kann M auf diesem Platz maximal $|K|$ verschiedene Konfiguration angenommen haben, daher reicht es aus, $|K|$ Berechnungsschritte von M zu untersuchen, um zu entscheiden, ob w akzeptiert wird. $|K| = |Q| \cdot u \cdot |\Gamma|^{s(n)} \cdot s(n) = O(c^{s(n)})$ für eine Konstante c.

Wenn man alle Berechnungen der Länge $\leq |K|$ mit Tiefensuche ausprobieren wollte, bräuchte man zuviel Platz. Definition eines Prädikates E (erreichbar):

$$E(k_1, k_2, t) :\Leftrightarrow k_1 \xrightarrow{*} k_2 \text{ in maximal } 2^t \text{ Schritten}$$

Dann gelten:

(i) $E(k_1, k_2, 0) \Leftrightarrow k_1 = k_2$ oder $k_1 \to k_2$

(ii) $E(k_1, k_2, t+1) \Leftrightarrow \exists k \in K : E(k_1, k, t)$ und $E(k, K_2, t)$

(iii) $w \in L \Leftrightarrow E(k_0, f, \log_2 |K|)$ mit k_0 ist die Startkonfiguration $q_0 w$

Wir berechnen nun Schrittweise die Prädikate, indem wir alle Konfigurationen $k \in K$ ausprobieren. Dazu sei eine Ordnung auf K gegeben. Dies dauert zwar sehr lange, liefert aber die gewünschte Platzschranke.

Algorithmus $E(k_1, k_2, t)$

```
if t = 0 then
    return k_1 = k_2 oder k_1 → k_2
κ = erste Konfiguration aus K.
repeat
    if E(k_1, κ, t − 1) und E(κ, k_2, t − 1) then
        return true
    κ = Nachfolgekonfiguration(κ)
until κ keine gültige Konfiguration mehr
return false
```

Platzbedarf: Pro Rekursionstiefe wird eine Konfiguration κ gespeichert mit $|\kappa| \in O(s(n))$, die maximale Tiefe ist $\log_2 |K| = O(\log_2(c^{s(n)})) = O(s(n))$, insgesamt also $O(s(n)^2)$. □

3.5. Satz: Es gilt: CFL $\subseteq$ DSPACE$((\log n)^2)$.

3.6. Satz: Das Wortproblem für det. 2-Wege Kellerautomaten ist in linearer Zeit lösbar.

4. Weitere Sprachklassen

4.1. Definition:

(i) Eine *Matrixgrammatik* ist ein 4-Tupel $G = (V, \Sigma, M, S)$ mit Matrizen $M \subseteq (((V \cup \Sigma)^+ \setminus \Sigma^*) \times (V \cup \Sigma)^*)^*$. *Matrixgrammatik*
In jedem Ableitungsschritt wird eine Matrix ausgewählt, dann werden alle Produktionen der Matrix angewandt.

(ii) Seien $u, v \in (V \cup \Sigma)^*$.

$$u \to v :\Leftrightarrow \begin{cases} \exists m = [P_1 \to Q_1, \ldots, P_r \to Q_r] \in M \text{ und } \exists v_1, \ldots, v_{r+1} \in (V \cup \Sigma)^* \text{ mit} \\ v_1 = u, v_{r+1} = v \wedge \forall i \in \{1, \ldots, r\} : v_i = \alpha_i P_i \beta_i \text{ mit } \alpha_i, \beta_i \in (V \cup \Sigma)^* \\ \wedge v_{i+1} = \alpha_i Q_i \beta_i \end{cases}$$

$\overset{*}{\to}$ ist der transitive Abschluss von $\to$.

(iii) mit *Anwesenheitsüberprüfung*: Menge von Regeln, die übersprungen werden können, falls die linke Seite nicht vorkommt. *Anwesenheitsüberprüfung*

4.2. Definition: Eine Matrixgrammatik heißt kontextfrei (beschränkt, linear, ...), wenn alle in den Matrizen auftretenden Regeln kontextfrei (beschränkt, linear, ...) sind.

4.3. Definition:

(i) M: Klasse der ε-freien kontextfreien Matrixgrammatiken

(ii) M^{ε}: Klasse der kontextfreien Matrixgrammatiken

(iii) M_{ac}: Klasse der ε-freien kontextfreien Matrixgrammatiken mit Anwesenheitsüberprüfung (appearance check)

(iv) $\mathrm{M}_{ac}^{\varepsilon}$: Klasse der kontextfreien Matrixgrammatiken mit Anwesenheitsüberprüfung

4.4. Definition:

(i) Sei $G = (V, \Sigma, P, S)$ eine Grammatik. Sei $\mathrm{Bez}(P)$ eine Menge, sei $\varphi : \mathrm{Bez}(P) \to P$ surjektiv. Sei $D : v \overset{*}{\to} w$ eine Ableitung der Länge r mit $\exists v_1, \ldots, v_{r+1} : v = v_1 \wedge u = v_{r+1} \wedge \forall i \in \{1, \ldots, r\} : v_i \to v_{i+1}$.
$u \in \mathrm{Bez}(P)^*$ heißt *Kontrollwort* von $D :\Leftrightarrow \exists u_1, \ldots, u_{r+1} \in \mathrm{Bez}(P)^*, \alpha_1, \ldots, \alpha_r \in \mathrm{Bez}(P)$ mit *Kontrollwort*
(1) $\varphi(\alpha_i) = (v_i \to v_{i+1}$ und
(2) $\forall \beta \in \mathrm{Bez}(P) : u_i$ enthält $\beta, \varphi(\beta) = (P \to Q) \Rightarrow v_i$ enthält P nicht.

Kontrollsprache

(ii) Eine Grammatik mit *Kontrollsprache* besteht aus einer Grammatik $G = (V, \Sigma, P, S)$, einer Menge Bez(P) (Eigenschaften wie oben), einer Kontrollsprache $C \subseteq$ Bez(P) und einer Menge auszulassender Regeln $F \subseteq P$.

(1) $L_{ac}(G, C, F) := \{w \in \Sigma^* | \exists$ Ableitung $D : S \overset{*}{\to} w$ und $a \in C$ mit a ist Kontrollwort von $D\}$

(2) $L(G, C) := L_{ac}(G, C, \emptyset)$

4.5. Definition: Für $i, j \in \{0, 1, 2, 2-\varepsilon, 3\}$ ist

- $\mathcal{L}(i, j, 0) := \{L(G, C) | G$ ist Type i-Gramatik, C ist Typ j-Sprache$\}$
- $\mathcal{L}(i, j, 1) := \{L_{ac}(G, C, F) | G$ ist Type i-Gramatik, C ist Typ j-Sprache$\}$

4.6. Lemma: Es gelten folgende Beziehungen:

(i) $\mathcal{L}(i, j, k) \subseteq \mathcal{L}(i, j', k)$ für $(j, j') \in \{(0, 1), (1, 2), (2, 3), (2, 2-\varepsilon)\}$

(ii) $\mathcal{L}_i \subseteq \mathcal{L}(i, j, k)$

(iii) $\mathcal{L}_j \subseteq \mathcal{L}(3, j, k)$

(iv) $\mathcal{L}(i, j, k) \subseteq \mathcal{L}(i', j', k)$ für $j = j'$ oder $(j, j') \in \{(0, 1), (1, 2), (2, 3), (2, 2-\varepsilon)\}$ und $(i, i') \notin \{(1, 2), (1, 3)\}$

4.7. Satz:

(i) $\mathcal{L}(0, 3, 0) = \mathcal{L}(0, 3, 1) = \mathcal{L}_0$

(ii) $\mathcal{L}(1, 3, 0) = \mathcal{L}(1, 3, 1) = \mathcal{L}_1$

(iii) $\mathcal{L}(3, 3, 0) = \mathcal{L}(3, 3, 1) = \mathcal{L}_3$

(iv) $\mathrm{M} = \mathcal{L}(2-\varepsilon, 3, 0)$, $\mathrm{M}^\varepsilon = \mathcal{L}(2, 3, 0)$, $\mathrm{M}_{ac} = \mathcal{L}(2-\varepsilon, 3, 1)$, $\mathrm{M}^\varepsilon_{ac} = \mathcal{L}(2, 3, 1)$

(v) $\mathrm{M} \subseteq \mathcal{L}_1$, $\mathrm{M}_{ac} \subseteq \mathcal{L}_1$

(vi) $\mathrm{M}^\varepsilon_{ac} = \mathcal{L}_0$

4.8. Satz (1994): Sei $L \subseteq \{1\}^*$ mit $L \in \mathrm{M}^\varepsilon$. Dann gilt $L \in \mathcal{L}_3$.

programmierte Grammatik

4.9. Definition: Eine *programmierte Grammatik* ist ein 4-Tupel $G = (V, \Sigma, R, S)$ mit R endliche Menge von Tripeln mit Namen $r_1, \ldots, r_n$, $R \subseteq P \times \mathrm{Pot}(\{r_i\}) \times \mathrm{Pot}(\{r_i\})$ mit P endlicher Menge von Produktionen.

$w \in L(G) :\Leftrightarrow$

(i) $\exists$ Ableitung $S = \alpha_0 \to \cdots \to \alpha_r = w$ mit $\forall i \in \{1, \ldots, r\} : \alpha_i \in (V \cup \Sigma)^*$ und

(ii) $\forall i \in \{1 \ldots, r-1\} : \alpha_i \to \alpha_{i+1}$ werde mit Regel $r_i = (A_i \to \beta_i, \sigma_i, \varphi_i)$ durchgeführt und

(iii) Falls A_i in α_i auftritt, entsteht α_{i+1} aus α_i durch Anwendung von $A_i \to \beta$ und $r_{i+1} \in \sigma_i$ und

(iv) Falls A_i in α_i nicht auftritt, ist $\alpha_i = \alpha_{i+1}$ und $r_{i+1} \in \varphi_i$.

G heißt *programmierte Grammatik ohne Anwesenheitsüberprüfung*, falls gilt:

$(A \to \alpha, \sigma, \varphi) \in R \Rightarrow \varphi = \emptyset$.

Sprachklassen: P, P^{ε}, P_{ac}, $\mathrm{P}^{\varepsilon}_{ac}$

4.10. Satz:

(i) $\mathrm{M} = \mathrm{P}$

(ii) $\mathrm{M}^{\varepsilon} = \mathrm{P}^{\varepsilon}$

(iii) $\mathrm{M}_{ac} = \mathrm{P}_{ac}$

(iv) $\mathrm{M}^{\varepsilon}_{ac} = \mathrm{P}^{\varepsilon}_{ac}$

5. LR-Grammatiken

(vergl. [HU90])

Element

5.1. Definition: Für eine kontextfreie Gramatik $G = (V, \Sigma, P, S)$ ist ein *Element* eine Produktion mit einem Punkt irgendwo auf der rechten Seite. Seien $A \in V$, $\alpha, \beta \in (V, \Sigma)^*$.

$A \to \alpha \cdot \beta$ ist Element $\Leftrightarrow (A \to \alpha\beta) \in P$

5.2. Definition:

Rechtssatzform

(i) Eine *Rechtssatzform* ist eine Satzform[†], die mit einer Rechtsableitung[‡] abgeleitet werden kann.

Handle

(ii) Ein *Handle* einer Rechtssatzform γ ist eine Teilzeichenkette β mit $S \underset{r}{\overset{*}{\to}} \delta A w \underset{r}{\to} \delta \beta w$.

[†] $\alpha \in (V \cup \Sigma)^*$ heißt *Satzform*:$\Leftrightarrow S \overset{*}{\to} \alpha$

[‡] immer die am weitesten rechts stehende Variable wird ersetzt, Schreibweise $\underset{r}{\to}$

lebensfähiges Präfix

(iii) Ein *lebensfähiges Präfix* einer Rechtssatzform γ ist jedes Präfix von γ, das nicht weiter rechts endet als das rechte Ende eines Handles von γ.

5.3. Definition:

gültig

(i) Ein Element $A \to \alpha \cdot \beta$ heißt *gültig* für ein lebensfähiges Präfix γ, wenn eine Rechtsableitung $S \overset{*}{\underset{r}{\to}} \delta A w \underset{r}{\to} \delta\alpha w$ mit $\delta\alpha = \gamma$ existiert.

vollständig

(ii) Ein Element heißt *vollständig*, wenn der Punkt das am weitesten rechts stehende Symbol des Elements ist: $A \to \alpha\cdot$

5.4. Satz: Sei $G = (V, \Sigma, P, S)$ eine Grammatik. Sei $M = (Q, V \cup \Sigma, \delta, q_0, Q)$ ein nichtdeterministischer endlicher Automat mit $Q = \{A \to \alpha \cdot \beta \,|\, (A \to \alpha\beta) \in P\} \cup \{q_0\}$ und

(i) $\delta(q_0, \varepsilon) = \{S \to \cdot\alpha \,|\, (S \to \alpha) \in P\}$

(ii) $\delta(A \to \alpha \cdot B\beta, \varepsilon) = \{B \to \cdot\gamma \,|\, (B \to \gamma) \in P\}$

(iii) $\delta(A \to \alpha \cdot X\beta, X) = \{A \to \alpha X \cdot \beta \,|\, (A \to \alpha X\beta) \in P\}$.

Dann akzeptiert M die Menge der lebensfähigen Präfixe von G. Für ein lebensfähiges Präfix γ gilt: $\delta(q_0, \gamma) = \{A \to \alpha \cdot \beta \,|\, A \to \alpha \cdot \beta \text{ gültig für } \gamma\}$.

LR(0)-Grammatik

5.5. Definition: Eine Grammatik G ist eine *LR(0)-Grammatik*, falls gilt:

(i) Das Startsymbol steht nicht auf der rechten Seite einer Produktion.

(ii) Für jedes lebensfähige Präfix γ von G gilt: Wenn $A \to \alpha\cdot$ ein vollständiges, für γ lebensfähiges Präfix ist, dann ist kein anderes vollständiges Element und kein Element mit einem Terminalzeichen rechts des Punktes für γ gültig.

5.6. Satz: Sei G eine LR(0)-Grammatik. Dann gibt es einen Kellerautomaten M mit $L(G) = L(M)$.

5.7. Bemerkung:

(i) LR(0)-Sprachen haben die Präfixeigenschaft.

(ii) Sei G eine LR(0)-Gramatik. Dann enthält G keine Kreisregeln $A_1 \to A_2 \to \cdots \to A_1$.

5.8. Satz: Das Wortproblem für LR(0) ist in linearer Zeit entscheidbar.

5.9. Satz: Sei M ein DPDA. Dann gibt es eine LR(0)-Grammatik G mit $L(G) = N(M)$.

5.10. Definition: $\text{First}_k(y) := \{u \in \Sigma^k \big| y \underset{r}{\overset{*}{\to}} uz \text{ mit } z \in (V \cup \Sigma)^*\}$

5.11. Definition: Sei $G = (V, \Sigma, P, S)$ eine kontextfreie Grammatik. S trete nicht auf der rechten Seite einer Regel auf. G heißt *LR(k)-Grammatik* ($k \geq 0$), wenn aus *LR(k)-Grammatik*

(i) $S \underset{r}{\overset{*}{\to}} \alpha A w \underset{r}{\to} \alpha\beta w$, $w \in \Sigma^*$

(ii) $S \underset{r}{\overset{*}{\to}} \gamma B x \underset{r}{\to} \alpha\beta y$, $x \in \Sigma^*$

(iii) $\text{First}_k(w) = \text{First}_k(y)$

folgt, dass $\alpha = \gamma$, $A = B$, $x = y$ gilt.

5.12. Satz: Die Definitionen von LR(0)-Gramatik und LR(k)-Grammatik sind für $k = 0$ äquivalent, falls $\varepsilon \notin L(G)$.

6. Unentscheidbarkeit

6.1. Definition: Ein Algorithmus A entscheidet ein Problem P, wenn er für jedes Beispiel B des Problems die Lösung ausgibt. Ein Problem P heißt *entscheidbar*, falls es einen Algorithmus gibt, der P entscheidet. *entscheidbar*

6.2. Lemma: Sei P ein Problem für eine Sprachklasse, $i \leq j$. Wenn P für $\mathcal{L}_i$ entscheidbar $\Rightarrow$ P für $\mathcal{L}_j$ entscheidbar.

6.3. Definition: Eine Instantiierung (Beispiel) des *Postschen Korrespondezproblem* (PKP) hat die Form $A = w_1, \ldots, w_k$, $B = x_1, \ldots, x_k$ mit $w_i, x_i \in \Sigma^*$. *Postsches Korrespondezproblem*

(i) Die Instantiierung hat eine Lösung $:\Leftrightarrow \exists i_1, \ldots, i_n \in \{1, \ldots, k\} : w_{i_1} \ldots w_{i_n} = x_{i_1} \ldots x_{i_n}$.

(ii) Die Instantiierung des modifizierten PKPs hat eine Lösung $:\Leftrightarrow \exists i_1, \ldots, i_n \in \{1, \ldots, k\} : w_1 w_{i_1} \ldots w_{i_n} = x_1 x_{i_1} \ldots x_{i_n}$.

6.4. Lemma: Wenn das PKP entscheidbar wäre, wäre das MPKP entscheidbar, d.h. es gilt MPKP$\leq$PKP.

6.5. Satz: Das PKP ist unentscheidbar.

6.6. Satz: Seien L_1, L_2 kontextfreie Sprachen. Ob $L = L_1 \cap L_2$ kontextfrei ist, ist unentscheidbar.

7. Komplexität

7.1. Satz: Das Wortproblem für P_{ac} ist NP-hart.

7.2. Definition: $\mathrm{P}^1_{ac} := \{G \mid G$ kontextfreie programmierte Grammatik mit Anwesenheitsprüfung und G enthält nur Regeln $A \to \alpha$ mit $|\alpha| \geq 2\}$.

7.3. Satz: Das Wortproblem für P^1_{ac} ist NP-vollständig.

Literatur

[HU90] Hopcraft, John E. und Jeffrey D. Ullman: *Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie.* Addison-Wesley, 1990.

[MAK] Moll, Robert N., Michal A. Arbib und A.J. Kfoury: *An Introduction To Formal Language Theory.*

A. Übersicht über die Sprachklassen

Sprachklasse	Grammatik	Abgeschlossenheit	Obermengen
$\mathcal{L}_0$, $\mathcal{L}(0,3,0)$, $\mathcal{L}(0,3,1)$, $\mathrm{M}_{ac}^{\varepsilon}$, HOM($\mathcal{L}_1$), HOM(1DFA(2)),	allgemein, ε-freie kontextfreie Matrixgr. mit Anwesenheitsüberpr., progr. mit Anwesenheitsüberpr.	$\cup, \cap, h, h^{-1}$, nichtlösch. h, Kleene+	
$\mathcal{L}_1$, $\mathcal{L}(1,3,0)$, $\mathcal{L}(1,3,1)$, NSPACE(n), NLBA	kontextsensitiv, beschränkt	$\bar{}, \cup, \cap, h^{-1}$, nichtlösch. h, Kleene+	$\mathcal{L}_0$
$\mathcal{L}(2-\varepsilon,3,1)$, M_{ac}, P_{ac}	ε-freie kontextfreie Matrixgr. mit Anwesenheitsüberpr., prog. mit mit Anwesenheitsüberpr.		$\mathcal{L}_1$
$\mathcal{L}(2,3,0)$, M^{ε}, P^{ε}	kontextfreie Matrixgr., programmierte		
$\mathcal{L}(2-\varepsilon,3,0)$, M, P	ε-freie kontextfreie Matrixgr., programmierte		$\mathcal{L}_1$
$\mathcal{L}_2$, NPDA	kontextfrei	$\cup, h, h^{-1}$, nichtlösch. h, Kleene+	$\mathcal{L}_1$
eind. CFL	eindeutig kontextfrei		$\mathcal{L}_2$
DCFL, DPDA			eind. CFL
LIN, 1-turn NPDA	linear		$\mathcal{L}_2$
$\mathcal{L}_3$, $\mathcal{L}(3,3,0)$, $\mathcal{L}(3,3,1)$, 1DFA(1), 1NFA(1), 2DFA(1), 2NFA(1)	regulär	$\bar{}, \cup, \cap, h, h^{-1}$, nichtlösch. h, Kleene+	DCFL, LIN

Index

Übergangsrelation 5

A

abgeschlossen
    unter Homomorphismus 3
allgemeine Sprachen 2
Alphabet 5

B

Bandalphabet 5
bandkonstruierbar 7
beschränkt 1
Blanksymbol 5

C

Chomsky-Normalform 9

D

deterministisch 5
    Kellerautomat 6

E

eindeutig 15
Element 25
Endlicher Automat 6
entscheidbar 27
erzeugte Sprache 1

G

Grammatik 1

H

Handle 25
Homomorphismus 2

I

Immerman 8

K

Kelleralphabet 6
Kellerautomat 6
Kellerendesymbol 6
kontextfrei 1
kontextfreie Sprachen 2
kontextsensitiv 1
kontextsensitive Sprachen 2
Kontrollsprache 24
Kontrollwort 23

L

linear
    Grammatik 1
    Menge 11
linear beschränkter Automat 6
lineare Sprachen 2
LR(k)-Grammatik 27
LR(0)-Grammatik 26

M

Matrixgrammatik 23
mehrdeutig 15

N

nichtdeterministisch 5

P

Parikh
    Abbildung 11
    Satz von 11
Postsches Korrespondezproblem 27
Produktion 1
programmierte Grammatik 25

R

rechtslinear 1
Rechtssatzform 25
reduziert 9
reguläre Sprachen 2

S

Savitch 21
semilinear 11
Sprachklassen 2
    Übersicht 10
Startsymbol 1
Szelepcsényi 8

T

Terminalsymbol 1
Turingmaschine 5

V

Variable 1