Automatentheorie und formale Sprachen

Prof. Dr. K. Indermark
Mitgeschrieben von Christian Haselbach
Korrektur gelesen von Sebastian Funk

Sommer-Semester 99

Vorwort

Dies ist eine Mitschrift zu der Vorlesung Automatentheorie und formale Sprachen im Sommersemester 1999. (http://www-i2.informatik.rwth-aachen.de/Vorlesung/ATFS99/) Sie ist noch nicht fertig! (Geht ja auch nicht, das Semester läuft ja noch). Es ist nicht (oder nur sehr schwer) möglich, alles so zu TEXen, wie es Herr Prof. Dr. Indermark anschreibt, was nicht an seiner Tafel-Anschrift sondern an den Unterschieden zwischen Tafel/Kreide und LaTeX liegt. Wir geben uns mühe, alles so korrekt wie möglich zu halten, aber Fehler können immer passieren. Diese Mitschrift soll auf gar keinen Fall die Vorlesung ersetzen. Herr Prof. Dr. Indermark erzählt noch vieles mehr, neben dem was er anschreibt, was auch hilfreich zum Verständnis des Stoffes ist.

Dieses Skript soll Hörern der Vorlesung eine übersichtliche Mitschrift mit Inhaltsangabe und Index geben, um zum Beispiel Begriffe schneller zu finden. Außerdem soll es Studenten, die die Vorlesung erst noch hören wollen oder die es nur so interessiert, einen Einblick in dieses Fach geben.

Notation

Es gibt leichte Unterschiede in der Notation: In der Vorlesung wird für eine Folgerung $\curvearrowright$ benutzt, hier aber $\Longrightarrow$. Für die Ableitungsrelation wird in der Vorlesung eben dieses $\Longrightarrow$ benutzt, hier wird dafür dann eben $\curvearrowright$ benutzt.

Weitere Informationen: http://www.kawo1.rwth-aachen.de/~kritanus/

Inhaltsverzeichnis

Kapitel 1

Alphabete, Wörter, Sprachen

Zeichenreihen als Grundobjekte der Informatik.

1. Grundbegriff:

Σ Alphabet, nicht-leere endliche Menge. $a \in \Sigma$ Buchstabe, Character, Symbol.

2. Grundbegriff:

Σ^* Menge der Wörter über Σ.

$$\Sigma^* := \{a_1 a_2 \dots a_n | n \in N, a_i \in \Sigma\}$$

Wörter, Zeichenreihen, Zeichenketten, Strings. Sonderfall: $n = 0$: das leere Wort, Bezeichnung: ε

Operationen auf Σ^*

- **Verkettung(Konkatenation)**

  $$\_ \cdot \_ : \Sigma^* \times \Sigma^* \to \Sigma^* (w, v) \mapsto w \cdot v := wv$$

  Es gilt:

  1. $\cdot$ ist assoziativ: $(u \cdot v) \cdot w = u \cdot (v \cdot w)$
  2. ε ist $\cdot$-neutral: $\varepsilon \cdot w = w \cdot \varepsilon = w$

  Sprechweise: $< \Sigma; \_ \cdot \_; \varepsilon >$ ist eine Halbgruppe mit 1 (Monoid).
  Bem.: freie Erzeugung

- **Länge eines Wortes** $w = a_1 a_2 a_3 \dots a_n \in \Sigma^*$:
  $|a_1 \dots a_n| := n$, falls alle $a_i \in \Sigma$
  also: $|\varepsilon| = 0$ und $|wv| = |w| + |v|$

- **Potenzen eines Wortes** $w \in \Sigma^*$**:**
  $w^0 := \varepsilon$
  $w^{n+1} := ww^n$

- **Spiegelbild eines Wortes**
  $\varepsilon^R := \varepsilon$
  $(wa)^R := a(w^R)$

3. Grundbegriff

$\mathfrak{P}(\Sigma^*)$ Menge der formalen Sprachen über Σ.

$$\mathfrak{P}(\Sigma^*) := \{L | L \subseteq \Sigma^*\}$$

Bsp.: ϕ, $\{\varepsilon\}$, $\{w_1, \ldots, w_n\}$, Σ^*

Operationen auf $\mathfrak{P}(\Sigma^*)$

- **boolesche Operationen:** $L_1 \cup L_2$, $L_1 \cap L_2$, $\bar{L} := \Sigma^* \backslash L$
- **Komplexprodukt:** $L_1 L_2 := \{w_1 w_2 | w_i \in L_i\}$
- **Potenzen einer Sprache:**
  $L^0 := \{\varepsilon\}$
  $L^{n+1} := LL^n$
- **Stern einer Sprache (Iteration, Repetition):**
  $L^* := \bigcup_{n \in \mathbb{N}} L^n \qquad L^+ := LL^* = \bigcup_{n=1}^{\infty} L^n$
- **Spiegelbild einer Sprache:** $L^R := \{w^R | w \in L\}$
- **reguläre Operationen:**
  $L_1 \cup L_2$, $L_1 L_2$, L^*

Kapitel 2

Reguläre Ausdrücke und endliche Automaten

2.1 Reguläre Ausdrücke

Ein regulärer Ausdruck ist eine endliche Beschreibung unendlicher Sprachen. Wesentliches Hilfsmittel ist der Stern *

Definition (Syntax): Sei Σ ein Alphabet. Die Menge $RA(\Sigma)$ **der regulären Ausdrücke über** Σ ist induktiv definiert durch

1. $\Lambda \in RA(\Sigma)$
2. $a \in RA(\Sigma)$ für jedes $a \in \Sigma$
3. $(\alpha \vee \beta) \in RA(\Sigma)$ falls $\alpha, \beta \in RA(\Sigma)$
4. $(\alpha \cdot \beta) \in RA(\Sigma)$ falls $\alpha, \beta \in RA(\Sigma)$
5. $(\alpha^*) \in RA(\Sigma)$ falls $\alpha \in RA(\Sigma)$

Vereinfachende Schreibweise:

- Präzedenzregeln um Klammern zu sparen:

  * bindet stärker als $\cdot$

  $\cdot$ bindet stärker als $\vee$
- Das $\cdot$ wird unterdrückt

Beachte:
$RA(\Sigma)$ ist selbst eine formale Sprache.
$RA(\Sigma) \subseteq (\Sigma \cup \{\Lambda, (,), \cdot, \vee, ^* \})^*$

Definition (Semantik) Ein regulärer Ausdruck beschreibt eine formale Sprache über Σ:

- $[\![\_]\!] : RA(\Sigma) \longrightarrow \mathfrak{P}(\Sigma^*)$

- $[\![\Lambda]\!] := \phi$
- $[\![(\alpha \vee \beta)]\!] := [\![\alpha]\!] \cup [\![\beta]\!]$
- $[\![(\alpha \cdot \beta)]\!] := [\![\alpha]\!][\![\beta]\!]$
- $[\![(\alpha^*)]\!] := [\![\alpha]\!]^*$

Die so beschreibbaren Sprachen heißen **regulär**.

Definition: Die Klasse $REG(\Sigma)$ der regulären Sprachen über Σ ist induktiv definiert durch:

- $\phi, \{a\} \in REG(\Sigma)$ für alle $a \in \Sigma$
- $L, L' \in REG(\Sigma) \longrightarrow L \cup L', LL', L^* \in REG(\Sigma)$

Die Menge WP der **WHILE$_0$-Programme**[1] ist induktiv aufgebaut über den Mengen
$A := \{X_i := X_i + 1, \quad X_i := X_i - 1 | i \in \mathbb{N}\}$ und
$B := \{X_i > 0 | i \in \mathbb{N}\}$ durch

1. $A \subseteq WP$
2. $P, Q \in WP \Longrightarrow$ begin $P; Q$ end $\quad \in WP$
3. $b \in B$, $P, Q \in WP \Longrightarrow$ if b then P else $Q \quad \in WP$
4. $b \in B$, $P \in WP \Longrightarrow$ while b do $P \quad \in WP$

Sei $\Sigma_{\mathsf{PATH}} := A \cup (B \times \{\mathsf{true}, \mathsf{false}\})$ und zur Vereinfachung $b_{\mathsf{true}} := (b, \mathsf{true})$; $b_{\mathsf{false}} := (b, \mathsf{false})$

Für $P \in WP$ definieren wir die **formale Pfadsprache** $L_P \subseteq \Sigma^*_{\mathsf{PATH}}$

1. $L_a := \{a\}$ für $a \in A$
2. $L_{\text{begin } P;Q \text{ end}} := L_P L_Q$
3. $L_{\text{if } b \text{ then } P \text{ else } Q} := \{b_{\mathsf{true}}\} L_P \cup \{b_{\mathsf{false}}\} L_Q$
4. $L_{\text{while } b \text{ do } P} := (\{b_{\mathsf{true}}\} L_P)^* \{b_{\mathsf{false}}\}$

Es folgt: Beschreibung von L_P durch $\alpha_P \in RA(\Sigma_{\mathsf{PATH}})$:

1. $\alpha_a = a$
2. $\alpha_{\text{begin } P;Q \text{ end}} := \alpha_P \alpha_Q$
3. $\alpha_{\text{if } b \text{ then } P \text{ else } Q} := b_{\mathsf{true}} \alpha_P \vee b_{\mathsf{false}} \alpha_Q$
4. $\alpha_{\text{while } b \text{ do } P} := (b_{\mathsf{true}} \alpha_P)^* b_{\mathsf{false}}$

Beachte: Analogie zwischen WP und $RA(\Sigma_{\mathsf{PATH}})$. Jedoch: Abstraktion von Anweisungs- und Bedingungssemantik.

Definition: $P \widetilde{\mathsf{PATH}} Q \iff L_P = L_Q$ (Pfadäquivalent)

Folgerung: $P \widetilde{\mathsf{PATH}} Q \implies P \sim Q$ (d.h. $f_P = f_Q$). Beachte: $P \widetilde{\mathsf{PATH}} Q$ ist durch endliche Automaten entscheidbar, während $P \sim Q$ unentscheidbar ist.

[1] Bezeichnung bei Prof. Thomas, Berechenbarkeit und Komplexität (WS 98/99)

Standard Probleme für $RA(\Sigma)$:

1. **Wortproblem (matching Problem):**
   Gilt $w \in [\![\alpha]\!]$ für $w \in \Sigma^*$?

2. **Äquivalenz-Problem:**
   Gilt $[\![\alpha]\!] = [\![\beta]\!]$ für $\alpha, \beta \in RA(\Sigma)$?

3. **Leerheits-Problem:**
   Gilt $[\![\alpha]\!] = \phi$ für $\alpha \in RA(\Sigma)$?

All diese Probleme sind mit endlichen Automaten entscheidbar.

2.2 Deterministische endliche Automaten

Definition: Seien Q und Σ nicht-leere, endliche Mengen, $q_0 \in Q$, $F \subseteq Q$ und $\delta : Q \times \Sigma \longrightarrow Q$. Dann heißt $\mathfrak{A} =< Q, \Sigma, \delta, q_0, F >$ ein deterministischer endlicher Automat über Σ mit der **Zustandsmenge** Q, dem **Eingabealphabet** Σ, der **Transitionsfunktion** δ, dem **Anfangszustand** q_0 und der **Endzustands-Menge** F.

Bezeichnung: $\mathfrak{A} \in DFA(\Sigma)$, $\mathfrak{A} \in DFA$ Darstellung $\mathfrak{A}$ durch **Transitionstafel** und **Zustandsgraphen**.

Definition: $\mathfrak{A} =< Q, \Sigma, \delta, q_0, F > \in DFA(\Sigma)$ bestimmt die **erweiterte Transitionsfunktion** $\bar{\delta} : Q \times \Sigma^* \text{ mit } \bar{\delta}(q, \epsilon) := q$
$\bar{\delta}(q, wa) = \delta(\bar{\delta}(q, w), a)$
beziehungsweise
$\bar{\delta}(q, aw) = \bar{\delta}(\delta(q, a), w)$

Ziel: $\mathfrak{L}(\Sigma, DFA) = REG(\Sigma)$
Hilfsmittel: nicht-deterministische Automaten.

2.3 Nicht-deterministische Automaten

Definition: Seien Q, Σ, q_0 und F wie bei einem $\mathfrak{A} \in DFA$, ferner $\Sigma_\varepsilon := \Sigma \cup \{\varepsilon\}$ und $\delta : Q \times \Sigma_\varepsilon \longrightarrow \mathfrak{P}(Q)$. Dann heißt $\mathfrak{A} =< Q, \Sigma, \delta, q_0, F >$ ein **nicht-deterministischer, endlicher Automat über** Σ

Bezeichnung: $NFA(\Sigma)$, NFA
Es folgt: $DFA(\Sigma) \subseteq NFA(\Sigma)$

Hinweis: Worttransitionen sind durch Zwischenzustände simulierbar.

Semantik eines $\mathfrak{A} \in NFA(\Sigma)$**:** Menge der **Konfigurationen** von $\mathfrak{A} : Q \times \Sigma^*$
Transitionen für alle $q, q' \in Q, w \in \Sigma^*, a \in \Sigma$ definieren wir

- Σ-Transitionen: $(q, aw) \vdash (q', w) \Longleftrightarrow q' \in \delta(q, a)$
- ε-Transitionen: $(q, w) \vdash (q', w) \Longleftrightarrow q' \in \delta(q, \epsilon)$

Dann ist die **durch $\mathfrak{A}$ erkannte Sprache** definiert als

$$L(\mathfrak{A}) := \{w \in \Sigma^* | (q_0, w) \overset{*}{\vdash} (q, \varepsilon) \text{ mit } q \in F\}$$

Definition: Sei $\mathfrak{A} \in NFA(\Sigma)$. Der **Potenzmengenautomat** $\mathfrak{A}_P :=< Q_P, \Sigma, \delta_P, q_{0P}, F_P > \in DFA(\Sigma)$ ist definiert durch:

- $Q_P := \{T \subseteq Q | \exists w \in \Sigma^* : \forall q \in T : (q_0, w) \overset{*}{\vdash} (q, \varepsilon)\}$
- $q_{0P} := \{q \in Q | (q_0, \varepsilon) \overset{*}{\vdash} (q, \varepsilon)\}$
- $F_P := \{T \in Q_P | T \cap F \neq \phi\}$
- $\delta_P := Q_P \times \Sigma \longrightarrow Q_P$
  $\delta_P(T, a) := \{q' \in Q | (q, a) \overset{*}{\vdash} (q', \varepsilon) \text{ für ein } q \in T\}$

Beachte: $T \in Q_p \Longrightarrow \exists w \in \Sigma^* \quad T = \{q' \in Q | (q_0, w) \overset{*}{\vdash} (q, \varepsilon)\} \Longrightarrow \delta_q(T, a) = \{q' \in Q | (q_0, wa) \overset{*}{\vdash} (q', \varepsilon)\} \in Q_p$

Lemma: Sei $\mathfrak{A} \in NFA(\Sigma)$. Dann gilt: $L(\mathfrak{A}) = L(\mathfrak{A}_p)$

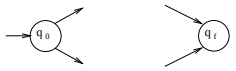
Beweis: Zunächst zeigen wir, daß für alle $w \in \Sigma^*$ und $q \in Q$ gilt: $q \in \bar{\delta_p}(q_{0_p}, w) \Longleftrightarrow (q_0, w) \overset{*}{\vdash} (q, \varepsilon)$

- $w = \varepsilon$
  $q \in \underbrace{\bar{\delta_p}(q_{0_p}, \varepsilon)}_{=q_{0_p}} \Longleftrightarrow q \in q_{0_p} \Longleftrightarrow (q_0, \varepsilon) \overset{*}{\vdash} (q, \varepsilon)$
- $w = w'a$
  $q \in \bar{\delta_p}(q_{0_p}, w'a) = \delta_p(\bar{\delta_p}(q_{0_p}, w'), a) \Longleftrightarrow$
  $\exists q' \in \bar{\delta}_p(q_{0_p}, w') : (q', a) \overset{*}{\vdash} (q, \varepsilon) \Longleftrightarrow$
  $\exists q' \in Q : (q_{0_p}, w'a) \overset{*}{\vdash} (q', a) \overset{*}{\vdash} (q, \varepsilon) \Longleftrightarrow$
  $(q_0, w'a) \overset{*}{\vdash} (q, \varepsilon)$
  Daraus folgt die Behauptung: $w \in L(\mathfrak{A}) \Longleftrightarrow \exists q \in F : (q_0, w) \overset{*}{\vdash} (q, \varepsilon)$
  $q \in \bar{\delta_p}(q_{0_p}, w)$ und $q \in F$
  $\bar{\delta_p}(q_{0_p}, w) \in F \Longleftrightarrow w \in L(\mathfrak{A}_p)$

2.4 Synthese und Analyse endlicher Automaten

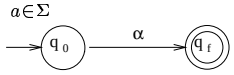
Synthese: Konstruiere für $\alpha \in RA(\Sigma)$ einen äquivalenten $\mathfrak{A}(\alpha) \in NFA$, das heißt: $[\![\alpha]\!] = L(\mathfrak{A}(\alpha))$

Der Algorithmus von Thompson
Idee: $\mathfrak{A}(\alpha)$ hat genau einen Endzustand q_f, wobei q_0 Quelle ist und q_f Senke ist (im Zustands Graphen).

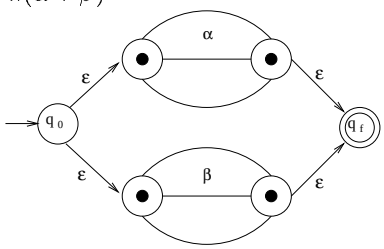


$\mathfrak{A}(\Lambda) :=$

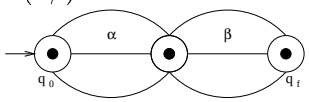
$\mathfrak{A}(a) :=$
$a \in \Sigma$



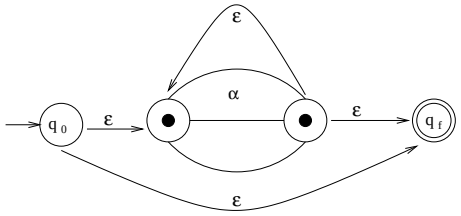
$\mathfrak{A}(\alpha \vee \beta) :=$



$\mathfrak{A}(\alpha\beta) :=$



$\mathfrak{A}(\alpha^*) :=$



Offensichtlich gilt für jedes $\alpha \in RA(\Sigma) : [\![\alpha]\!] = L(\mathfrak{A}(\alpha))$

Korollar: $REG(\Sigma) \subseteq L(\Sigma, DFA)$

Analyse: Konstruiere für $\mathfrak{A} \in DFA(\Sigma)$ ein $\alpha(\mathfrak{A}) \in RA(\Sigma)$ mit $[\![\alpha(\mathfrak{A})]\!] = L(\mathfrak{A})$
$\mathfrak{A} =< Q, \Sigma, \delta, q, F > \in DFA(\Sigma)$ und $Q = \{q_1, \ldots, q_n\}$
Für $i, j \in \{1, \ldots, n\}$ und $k \in \{0, 1, \ldots, n\}$ definieren wir:

$$W_{ij}^k := \{w \in \Sigma^* \quad | \text{w überführt } q_i \text{ in } q_j \text{ ohne Benutzung von } q_{k+1}, \ldots, q_n \text{ als Zwischenzustände}\}$$

Dann gilt: $L(\mathfrak{A}) = \bigcup_{q_j \in F} W_{ij}^n$
Somit genügt der Nachweis der Regularität der Sprache W_{ij}^k. Induktion über k:

1. $k = 0 :\ W_{ij}^0 \subseteq \Sigma_\varepsilon$ ist regulär, $\{\varepsilon\} = [\![\Lambda^*]\!]$
2. $k - 1 \to k :\ W_{ij}^k = W_{ij}^{k-1} \cup W_{ik}^{k-1} \left(W_{kk}^{k-1}\right)^* W_{kj}^{k-1}$

Korollar(Satz von Kleene): $L(\Sigma, DFA) = REG(\Sigma)$

Korollar: $REG(\Sigma)$ ist auch abgeschlossen unter $\cap, -$.
Beweisidee:

1. Komplement: $F' := Q \backslash F$
2. $L_1 \cap L_2 := \overline{\bar{L_1} \cup \bar{L_2}}$

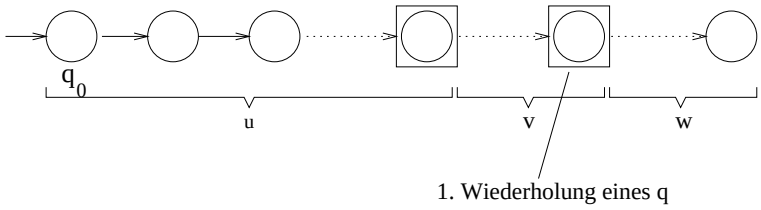
2.5 Das Pumping-Lemma

Hilfsmittel zum Nachweis nicht regulärer Sprachen.

Satz(Pumping-Lemma, Iterationslemma)
Sei $L \in REG(\Sigma)$. Dann existiert $k \in \Sigma$, so daß für jedes $x \in L$ mit $|x| \geq k$ eine Zerlegung $x = uvw$ mit folgenden Eigenschaften existiert:

1. $|v| \geq 1$
2. $|uv| \leq k$
3. $uv^iw \in L$ für alle $i \in \mathbb{N}$, insbesondere für $i = 0$

Beweis: Sei $\mathfrak{A} \in DFA(\Sigma)$ mit $L(\mathfrak{A}) = L$ und $k = |Q|$. Außerdem sei $x \in L$ mit $|x| \geq k$.
Erkennung von x in $\mathfrak{A}$:



Seien ◯ das erste Wiederholung-Paar von Zuständen, so folgen (1.)-(3.).

Beachte: Das Pumping-Lemma beschreibt eine notwendige, aber nicht hinreichende Eigenschaft regulärer Sprachen. Daher eignet es sich sehr gut um nicht-reguläre Sprachen nachzuweisen.

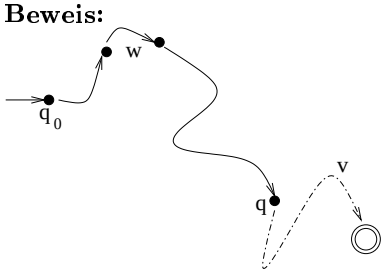
Beispiel: $L = \{a^nb^n | n \geq 1\} \in REG(\{a, b\})$
Beweis: Angenommen $L \in REG$. Dann existiert $k \in \mathbb{N}$ mit den Eigenschaften des Pumping-Lemmas ("Pumping-Index"). Für $x = a^kb^k$ muß eine Zerlegung existieren $a^kb^k = uvw$ mit $v \neq \varepsilon$ und $|uv| \leq k$, also $v \in \{a^i | i > 0\}$ und $uw = a^{k-|v|}b^k \in L$. Widerspruch.

2.6 Zustandsreduktion endlicher Automaten

Ziel: Konstruktion endlicher Automaten mit minimaler Zustandsanzahl.

Definition: Für $L \subseteq \Sigma^*$ und $w \in \Sigma^*$ heißt $d_w(L) := \{v \in \Sigma^* | wv \in L\}$ **die Ableitung von** L **nach** w

Lemma: Sei $L = L(\mathfrak{A})$ für $\mathfrak{A} =< Q, \Sigma, \delta, q, F > \in DFA(\Sigma)$. Dann gilt für $D(L) := \{d_w(L) | w \in \Sigma^*\}$: $|D(L)| \leq |Q|$
Beweis:



Für $q \in Q$ bezeichne $L(q) := L(< Q, \Sigma, \delta, q, F >)$. Dann gilt: $d_w(L) = d_w(L(q_0)) = L(\bar{\delta}(q_0, w))$.

Definition: Der **Ableitungsautomat** $\mathfrak{A}_L :=< D(L), \Sigma, \delta, q_0, F >\in DFA(\Sigma)$ ist für $L \in REG$ definiert durch

- $q_0 := d_\varepsilon(L) = L$
- $F := \{d_w(L) | w \in L\}$ dh. $\varepsilon \in d_w(L)$
- $\delta(d_W(L), a) := d_{wa}(L)$

Beachte: $d_w(L) = d_v(L) \Longrightarrow d_{wa}(L) = d_{va}(L)$

Lemma: $L(\mathfrak{A}_L) = L$
Beweis: $w \in L(\mathfrak{A}_L) \Longleftrightarrow \bar{\delta}(q_0, w) \in F \Longleftrightarrow d_w(L) \in F \Longleftrightarrow w \in L$.

Korollar:

- Der Ableitungsautomat ist Zustandsminimal.
- Für $L \subseteq \Sigma^*$ gilt: $L \in REG \Longleftrightarrow |D(L)| < \infty$.

Zustandsreduktion: Konstruktion eines Zustandsminimalen Automaten aus einem gegebenen Automaten durch

- Weglassen nicht erreichbarer Zustände und
- Verschmelzen äquivalenter Zustände.

Dann heißt:

- $q \in Q$ erreichbar $:\Longleftrightarrow \exists w \in \Sigma^* : \bar{\delta}(q_0, w) = q$
- $q_1 \sim q_2$(äquivalent) $:\Longleftrightarrow L(q_1) = L(q_2)$

Definition: Für $\mathfrak{A} =< Q, \Sigma, \delta, q_0, F >\in DFA(\Sigma)$ ist der **Faktorenautomat**
$\mathfrak{A}/\sim :=< \tilde{Q}, \Sigma, \tilde{\delta}, \tilde{q_0}, \tilde{F} >\in DFA(\Sigma)$ wie folgt definiert:
Für $q \in Q$ sei $[q] := \{q' | q \sim q'\}$.

- $\tilde{Q} := \{[\bar{\delta}(q_0, w)] | w \in \Sigma^*\}$
- $\tilde{q_0} := [q_0]$
- $\tilde{F} := \{[q] | q \in F\}$
- $\tilde{\delta}([q], a) := [\delta(q, a)]$

Beachte: $q \sim q' \implies L(q) = L(q') \implies \delta(q, a) \sim \delta(q', a)$

Lemma: Für $\mathfrak{A} \in DFA(\Sigma)$ gilt: $\mathfrak{A}/\sim \; = \mathfrak{A}_{L(\mathfrak{A})}$ (bis auf Zustandsnamen).
Insbesondere ist $\mathfrak{A}/\sim$ äquivalent zu $\mathfrak{A}$ und es folgt: Zustandsminimale Automaten sind bis auf Isomorphie eindeutig bestimmt.
Beweis: Sei $\beta : \tilde{Q} \longrightarrow P(L(\mathfrak{A}))$ definiert durch $\beta([\bar{\delta}(q_0, w)]) := d_w(L(\mathfrak{A}))$

- β ist unabhängig vom Repräsentanten $w \in \Sigma^*$:
  $\bar{\delta}(q_0, v) \sim \tilde{\delta}(q_0, w) \implies L(\tilde{\delta}(q_0, w)) = L(\tilde{\delta}(q_0, v)) \implies d_w(L(\mathfrak{A})) = d_v(L(\mathfrak{A}))$

- β ist bijektiv, weil die obige Folgerung umkehrbar ist.
- β ist strukturerhaltend:
  - $\beta([q_0]) = \beta([\bar{\delta}(q_0, \varepsilon)]) = d_\varepsilon(L(\mathfrak{A}))$
  - $\bar{\delta}(q_0, w) = q \in F: \ \beta([\bar{\delta}(q_0, w)]) = d_w(L(\mathfrak{A}))$ Endzustände des Ableitungs-Automaten.
  - $\tilde{\delta}([q], a) = [\delta(q, a)] \Longrightarrow \tilde{\delta}([\bar{\delta}(q_0, w)], a) = [\bar{\delta}(q_0, wa)]$
    $\Longrightarrow \delta(d_w(L(\mathfrak{A})), a) = d_{wa}(L(\mathfrak{A}))$

Verfahren zur Zustandsminimierung

- Weglassen nicht erreichbarer Zustände.
- Verschmelzen äquivalenter Zustände.

Definition
Sei $\mathfrak{A} =< Q, \Sigma, \delta, q_0, F > \in DFA$ und jeder Zustand erreichbar. Ferner $k \in \mathbb{N}$; $q_1, q_2 \in Q$. Dann sagt man:

- q_1 k-äquivalent q_2 $(q_1 \overset{k}{\sim} q_2) :\Longleftrightarrow \forall w \in \Sigma^*, |w| \leq k: \ w \in L(q_1) \Longleftrightarrow w \in L(q_2)$
- $\mathfrak{A}$ induziert die Abbildung $r^k : Q^2 \longrightarrow \{0, 1\}$ mit $r^k(q_1, q_2) = 1 :\Longleftrightarrow q_1 \overset{k}{\sim} q_2$.
  Diese können als Matrizen $R^k = (r^k(q_i, q_j))_{i,j=1}^n$ mit $n = |Q|$ dargestellt werden.

Beachte: $r^k(q_i, q_j) = r^k(q_j, q_i)$

Berechnung der k-Äquivalenz-Matrizen

- $q_1 \overset{0}{\sim} q_2 \iff (q_1 \in F \Longleftrightarrow q_2 \in F)$
- $q_1 \overset{k+1}{\sim} q_2 \iff q_1 \overset{0}{\sim} q_2$ und $\delta(q_1, a) \overset{k}{\sim} \delta(q_2, a)$ für alle $a \in \Sigma$

Lemma:
Es gibt ein $k \in \mathbb{N}$, so daß für alle $n \in \mathbb{N}$ $R^k = R^{k+n}$ gilt.

Beweis:
Da $r^k(q_i, q_j) = 0 \implies r^{k+1}(q_i, q_j) = 0$, muß es ein k geben, mit $r^k = r^{k+1}$. Dann muß auch $r^k = r^{k+n}$ für alle $n \in \mathbb{N}$ gelten:
Sei $r^k = (q_1, q_2) = r^{k+1}(q_1, q_2) = 1$ also $q_1 \overset{k}{\sim} q_2$ und $q_1 \overset{k+1}{\sim} q_2$. Sei ferner $a_1 \ldots a_{k+2} \in L(q_1)$.
$\delta(q_1, a_1) \overset{k}{\sim} \delta(q_2, a_1)$, also auch $\delta(q_1, a_1) \overset{k+1}{\sim} \delta(q_2, a_1)$.
Somit $a_2 \ldots a_{k+2} \in L(\delta(q_2, a_1))$ und $a_1 \ldots a_{k+2} \in L(q_2)$.
Wegen Symmetrie folgt: $q_1 \overset{k+2}{\sim} q_2$.

Markierungsalgorithmus
Eingabe: $\mathfrak{A} =< Q, \Sigma, \delta, q_0, F > \in DFA$, jedes $q \in Q$ ist erreichbar.
Ausgabe: äquivalente Zustände.
Verfahren:

- Tabelle aller Zustandspaare $\{q, q'\}$ mit $q \neq q'$.
- Markierung aller Zustandspaare $\{q, q'\}$ mit $q \in F$ und $q' \notin F$ (oder umgekehrt).

- Für jedes unmarkierte Paar $\{q, q'\}$ und $a \in \Sigma$ teste, ob $\{\delta(q,a), \delta(q',a)\}$ markiert ist. Wenn ja: Markiere $\{q, q'\}$.
- Wiederhole letzten Schritt, bis keine Änderung mehr erfolgt.
- Unmarkierte Paare repräsentieren äquivalente Zustände.

Bemerkung: Implementation mit $O(|Q|^2)$ Zeitkomplexität möglich.

2.7 Entscheidbare Eigenschaften

Deterministische endliche Automaten (*DFA*)

- **Wortproblem:** $w \in L(\mathfrak{A})$? für $w \in \Sigma^*$ und $\mathfrak{A} \in DFA(\Sigma)$.
  Durch Eingabe in $|w|+1$ Schritten entscheidbar.
- **ϕ-Problem/Leerheitsproblem:** $L(\mathfrak{A}) = \phi$? für $\mathfrak{A} \in DFA(\Sigma)$.
  Durch Eingabe aller Worte w mit $|w| < |Q|$ entscheidbar.
  Beachte: $w \in L(\mathfrak{A}), |w| \geq |Q| \implies \exists v \in L(\mathfrak{A}),\ |v| < |w|$
  Verfahren: testen, ob F von q_0 erreichbar. Durchführung in $O(|Q|^2)$-Zeit. Bei geschickter Implementierung der Mengenoperationen auch besser.
- **$\sim$-Problem/Äquivalenzproblem:** $L(\mathfrak{A}) = L(\mathfrak{A}')$? für $\mathfrak{A}, \mathfrak{A}' \in DFA(\Sigma)$.
  Reduktion auf ϕ-Problem.
  $L(\mathfrak{A}) = L(\mathfrak{A}') \iff L(\mathfrak{A}) \subseteq L(\mathfrak{A}')$ und $L(\mathfrak{A}') \subseteq L(\mathfrak{A})$
  $\iff \underbrace{L(\mathfrak{A}) \cap \overline{L(\mathfrak{A}')}}_{\mathfrak{A}_1} = \phi$ und $\underbrace{L(\mathfrak{A}') \cap \overline{L(\mathfrak{A})}}_{\mathfrak{A}_2} = \phi$
  2 Produktautomaten mit $|Q| \cdot |Q'|$ Zuständen auf ϕ testen. $\sim$-Problem ist in $O(|Q|^2 \cdot |Q'|^2)$-Zeit lösbar.

Reguläre Ausdrücke, nicht-deterministische Automaten

Durch Transformation in *DFA*'s sind alle 3 Probleme entscheidbar. Aber der Aufwand steigt:

- **Wortproblem:** $O(|\alpha||w|)$-Zeit, $O(|\alpha||w|)$-Platz.
- **ϕ-Problem:** *NFA* wie *DFA* behandeln. *RA* $\mapsto$ *NFA* in $O(|\alpha|)$-Zeit mit $|Q| \leq 2|\alpha|$
- **$\sim$-Problem:** NP-Hart ($Pr \in NP \implies P \leq_P \sim -$Problem)

2.8 Endliche Automaten mit Ausgabe

Idee: Bei Zustandsübergängen erfolgt sequentielle Ausgabe.

Definiton: Sei $< Q, \Sigma, \delta, q_0, F > \in DFA$, Δ ein **Ausgabealphabet** (nicht-leere, endliche Menge) und $\lambda : Q \times \Sigma \to \Delta$ eine **Ausgabefunktion**. Dann heißt $\mathfrak{A} =< Q, \Sigma, \Delta, q_0, \delta, \lambda >$ ein **Mealy-Automat**. $\mathfrak{A}$ berechnet die **sequentielle Transformation** $f_{\mathfrak{A}} : \Sigma^* \longrightarrow \Delta^*$
mit $f_{\mathfrak{A}}(\varepsilon) := \varepsilon$
und $f_{\mathfrak{A}}(wa) := f_{\mathfrak{A}}(w)\lambda(\bar{\delta}(q_0, w), a)$

2.9 Automaten als abstrakte Programme

GOTO-Programme

GOTO-Programme sind aufgebaut über den Mengen
$A := \{X_i := X_i + 1,\ X_i := X_i - 1 \,|\, i \in \mathbb{N}\}$ (Anweisungen) und
$B := \{x_i = 0 | i \in \mathbb{N}\}$ (Bedingungen)
nämlich als Zeichenreihen der Form
$1 : \alpha_1; 2 : \alpha_2; \ldots k : \alpha_k$
mit $\alpha_i \in A \cup \{\mathsf{if}\, b\, \mathsf{then}\, j_0\, \mathsf{else}\, j_1 | b \in B, 1 \leq j_0, j_1 \leq k\}$ und $\alpha_k = \mathsf{STOP}$.

Sei $\Sigma_{\mathsf{PATH}} := A \cup (B \times \{\mathsf{true}, \mathsf{false}\})$.
Für ein GOTO-Programm $P = 1 : \alpha_1; \ldots k : \mathsf{STOP}$ definieren wir ein $\mathfrak{A}_P =< Q, \Sigma_{\mathsf{PATH}}, \delta, q_0, F > \in$ *DFA*

- $Q = \{1, \ldots, k, 0\}$
- $q_0 = 1$
- $F = \{k\}$
- $\delta(i, a) = i + 1$ falls $i : a; \in P$
  $\left.\begin{array}{l}\delta(i, b_{\mathsf{true}}) = j_0 \\ \delta(i, b_{\mathsf{false}}) = j_1\end{array}\right\}$ falls $i : \mathsf{if}\, b\, \mathsf{then}\, j_0\, \mathsf{else}\, j_1$
- $\delta(i, c) = 0$ sonst

Definition: $L_P := L(\mathfrak{A})$ heißt ***formale*** **Pfadsprache von** P $P \widetilde{\mathsf{PATH}} Q :\Longleftrightarrow L_P = L_Q$.

Lemma: $\underbrace{P \widetilde{\mathsf{PATH}} Q}_{\text{entscheidbar}} \implies \underbrace{P \sim Q}_{\text{unentscheidbar}}$ (aber nicht umgekehrt).

Parallele Programme, Verteilte System

Reduktion von Parallelität auf Nichtdet..
Für Aktionen a und b wird $a||b := (ab \vee ba)$
NFA's zur Modellierung verteilter Systeme. Beispiel: Deadlocks als Senkzustände.

Kapitel 3

Kontextfreie Grammatiken und Kellerautomaten

3.1 Kontextfreie Grammatiken

Definition: Seien N und Σ nicht-leere endliche Mengen mit $N \cup \Sigma = \phi$. Sei $P \subseteq N \times (N \cap \Sigma)^*$ mit $|P| < \infty$ und $S \in N$.
Dann heißt $G =< N, \Sigma, P, S >$ eine **kontextfreie Grammatik**. Bezeichnung: $G \in \mathit{CFG}(\Sigma)$ oder $G \in \mathit{CFG}$

Bezeichnung und Konventionen:

$A, B, C, \ldots \in N$	Nichtterminalsymbole
$a, b, c, \ldots \in \Sigma$	Terminalsymbole
$S \in N$	Startsymbol
$\ldots, X, Y, Z \in \chi := N \cup \Sigma$	Symbole
$\alpha, \beta, \gamma, \ldots \in \chi^*$	Satzformen
$u, v, w, \ldots \in \Sigma^*$	Terminalwörter
$A \longrightarrow \alpha := (A, \alpha) \in P$	Produktion, Regel

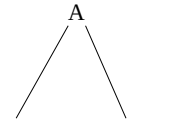
Definition: Sei $G =< N, \Sigma, P, S > \in \mathit{CFG}$ und $\pi = A \longrightarrow \alpha \in P$. π bestimmt eine **Ableitungsrelation** $\curvearrowright_\pi \subseteq \chi^* \times \chi^*$ mit $\beta_1 \curvearrowright_\pi \beta_2 :\Longleftrightarrow$ es existiert ein Kontext $\gamma_1, \gamma_2 \in \chi^*$, so daß $\beta_1 = \gamma_1 A \gamma_2$ und $\beta_2 = \gamma_1 \alpha \gamma_2$.
Dann heißt das Tripel $(\gamma_1, \pi, \gamma_2)$ auch **Ableitungsschritt**. G bestimmt die **Ableitungsrelation** $\curvearrowright_G \subseteq \chi^* \times \chi^*$ durch $\curvearrowright_G := \bigcup_{\pi \in p} \curvearrowright_\pi$ und damit **die von G erzeugte Sprache** $L(G) := \{w \in \Sigma^* | S \overset{*}{\curvearrowright}_G w\}$
$\overset{*}{\curvearrowright}_G := \bigcup_{n=0}^{\infty} \overset{n}{\curvearrowright}_G$ (reflexive und transitive Hülle).
Es folgt: $w \in L(G) \Longleftrightarrow \exists \alpha_0, \alpha_1 \ldots \alpha_n \in \chi^*$ und $\pi_1, \ldots, \pi_n \in P :\ S = \alpha_0 \curvearrowright_{\pi_1} \alpha_1 \ldots \curvearrowright_{\pi_n} \alpha_n = w$ $(n \geq 1)$

Bezeichnung: $\mathit{CFL}(\Sigma) := \{L \subseteq \Sigma^* | \exists G \in \mathit{CFG}(\Sigma) : L(G) = L\}$

Ableitungsbäume, Rechts- und Linksableitung, Ein- und Mehrdeutigkeit

Sei $G =< N, \Sigma, P, S > \in \mathit{CFG}$.
Darstellung von Ableitungen durch Bäume:

1. Eine Regel $\pi = A \longrightarrow X_1 \dots X_n$ bestimmt den Regelbaum

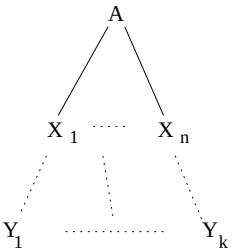


Spezialfall: 

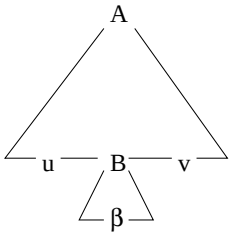
für n=0

2. Eine Ableitung $A \curvearrowright \alpha_1 \curvearrowright \alpha_1 \dots \curvearrowright \alpha_n$ bestimmt den **Ableitungsbaum** durch entsprechendes Verkleben der Regelbäume.
Definition durch Induktion über $n \in \mathbb{N}$

   - $n = 1$: Regelbaum
   - $n \longrightarrow n+1$: $A \curvearrowright \alpha_1 \dots \alpha_n$ habe den Ableitungsbaum:



   $\alpha_n \curvearrowright a_{n+1}$ mit dem Ableitungsschritt $\pi : B \longrightarrow \beta \quad (u, \pi, v)$.
   Dann entsteht der Ableitungsbaum von $A \curvearrowright \alpha_1 \dots \curvearrowright \alpha_{n+1}$ durch Anhängen des Regelbaums von π zwischen u und v.



Folgerung: Ein Ableitungsbaum repräsentiert eine Klasse von Ableitungsbäumen die sich nur in der Reihenfolge von Ableitungsschritten unterscheiden. Ein Ableitungsbaum repräsentiert die relevante syntaktische Struktur.

Definition: Eine Ableitung heißt **Rechtsableitung** (**Linksableitung**), wenn jeder Ableitungsschritt (α, π, β) ein **Rechtsableitungsschritt**, das heißt $\beta \in \Sigma^*$ (ein **Linksableitungsschritt**, das heißt $\alpha \in \Sigma^*$) ist.
Schreibweise: $\underset{l}{\curvearrowright}$ beziehungsweise $\underset{r}{\curvearrowright}$.

Folgerung: Ein Ableitungsbaum hat genau eine Rechts- und genau eine Linksableitung.

Definition:

- $G \in \mathit{CFG}(\Sigma)$ **eindeutig**:$\Longleftrightarrow$ zu jedem $w \in L(G)$ gibt es genau 1 Rechtsableitung $b \underset{r}{\curvearrowright} \alpha_1 \underset{r}{\curvearrowright} \dots \underset{r}{\curvearrowright} w$
- G **mehrdeutig**:$\Longleftrightarrow$ G nicht eindeutig.

3.2 Einseitig lineare Grammatiken

Definition: Sei $G =< N, \Sigma, P, S > \in$ *CFG*.

- Dann heißt G **linkslinear**, wenn für jedes $\pi = A \longrightarrow \alpha \in P$ gilt: $\alpha = Bw$ oder $\alpha = w$ für ein $B \in N$ und $w \in \Sigma^*$.
- Gilt stattdessen $\alpha = wB$ oder $\alpha = w$ für jedes $\pi \in P$, so heißt G **rechtslinear**.
- G **einseitig-linear**:$\Longleftrightarrow$ G linkslinear oder rechtslinear.

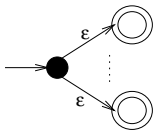
Ziel: $\{L(G)|G \text{ linkslinear}\} = \{L(G)|G \text{ rechtslinear}\} = \{L(G)|G \text{ einseitig-linear}\} = REG$

Satz: $\mathfrak{L}(\Sigma, NFA) = \{L \subseteq \Sigma^* | L = L(G), G \text{ rechtslinear}\}$
Beweis:
$\underline{\supseteq}$: Sei $G =< N, \Sigma, P, S >$ rechtslinear. Auffassung von G als $\mathfrak{A} =< Q, \Sigma, \delta, q_0, F > \in NFA$
$Q := N \cup \{q_0\}$, q_f neu, $\delta(A, w) \ni B$ falls $A \longrightarrow wB$ in G,$\delta(A, w) \ni q_f$ falls $A \longleftarrow w$, $q_0 := S$, $F := \{q_f\}$.
Offensichtlich: $L(G) = L(\mathfrak{A})$
Wortübergang durch Zwischenzustände beseitigen.
$\underline{\subseteq}$: Sei $\mathfrak{A} =< Q, \Sigma, \delta, q_0, F > \in DFA$. Auffassung von $\mathfrak{A}$ als $G =< N, \Sigma, P, S >$ rechtslinear.
$\underset{q}{\bigcirc} \xrightarrow{a} \underset{q'}{\bigcirc}$ Endzustände
$q \longrightarrow aq'$ $\quad q_f \mapsto q_f \longrightarrow \varepsilon$

Korollar: $REG(\Sigma) \subseteq CFL(\Sigma)$
Für $|\Sigma| \geq 2$ ist diese Inklusion echt, weil $\{a^n b^n | n \in \mathbb{N}\} = L(G)$ mit $G = (S \longrightarrow aSb \,|\, a \longrightarrow \varepsilon)$.

Bemerkung: $|\Sigma| = 1 \Longrightarrow REG(\Sigma) = CFL(\Sigma)$

Lemma: $L \in REG(\Sigma) \Longrightarrow L^R \in REG(\Sigma)$
Beweis: Zu $\mathfrak{A} \in DFA(\Sigma)$ konstruiere $\mathfrak{A}^R \in NFA(\Sigma)$ mit $L(\mathfrak{A}^R) = L(\mathfrak{A})^R$

Idee: ersetze $\underset{q}{\bullet} \xrightarrow{a} \underset{q'}{\bullet}$ durch $\underset{q}{\bullet} \xleftarrow{a} \underset{q'}{\bullet}$, $\longleftarrow \underset{q_0}{\bullet}$ durch $\underset{q_0}{\circledcirc}$ und ergänze diesen NFA durch 

für alle $q \in F$.

Korollar: $\{L \subseteq \Sigma^* | L = L(G), G \text{ rechtslinear}\} = \{L \subseteq \Sigma^* | L = L(G), G \text{ linkslinear}\}$
Beweis: L rechtslinear erzeugbar $\Longleftrightarrow$ L^R rechtslinear erzeugbar $\Longleftrightarrow$ L^{RR} linksliniear erzeugbar.

3.3 Normalformen von kontextfreien Grammatiken

Hilfsmittel: Vorgänger von Satzformen
$\alpha \curvearrowright \beta$ $\quad \alpha$ **direkter** Vorgänger von β.
$\alpha \overset{*}{\curvearrowright} \beta$ $\quad \alpha$ Vorgänger von β.

Definition: Sei $G =< N, \Sigma, P, S > \in CFG$ und $L \subseteq \chi^*$. Dann ist

1. die **direkte Vorgängermenge** von L definiert durch $\underline{\mathsf{Pre}}_G(L) := \{\alpha \in \chi^* | \exists \beta \in L : \alpha \curvearrowright_G \beta\}$.
2. der **Vorgängerabschluß** von L definiert durch $\underline{\mathsf{Pre}}^*_G(L) := \{\alpha \in \chi^* | \exists \beta \in L : \alpha \overset{*}{\curvearrowright} \beta\}$.

Lemma: Sei $G =< N, \Sigma, P, S >\in CFG$ und $L \subseteq \chi^*$. Dann gilt: $L \in REG(\chi) \Longrightarrow \underline{\mathsf{Pre}}^*(L) \subset REG(\chi)$.
Beweis: Sei $\mathfrak{A} =< Q, \chi, \delta, q_0, F >\in NFA$ mit $L(\mathfrak{A}) = L$. Konstruiere $\mathfrak{A}' =< Q, \chi, \delta', q_0, F >\in NFA$ durch hinzufügen folgender Transitionen: wenn $A \longrightarrow \alpha \in P$ und $\bar{\delta}(q, \alpha) \ni q'$ so $\delta'(q, A) \ni q'$. Ergänze δ um solche Transitionen, solange das möglich ist. Dieser Erweiterungsprozeß terminiert, weil Q und χ endlich sind. Die Korrektheit ist offensichtlich.

Definition: Sei $G \in CFG(\Sigma)$ und $A \in N$.

1. A heißt **produktiv**:$\Longleftrightarrow \exists w \in \Sigma^* : A \overset{*}{\curvearrowright} w$.
2. A heißt **erreichbar**:$\Longleftrightarrow \exists \alpha, \beta \in \chi : S \overset{*}{\curvearrowright} \alpha A \beta$.

Folgerung: A produktiv $\Longleftrightarrow A \in \underline{\mathsf{Pre}}^*_G(\Sigma^*)$
A erreichbar $\Longleftrightarrow S \in \underline{\mathsf{Pre}}^*_G(\chi^*\{A\}\chi^*)$

Definition: $G \in CFG(\Sigma)$ heißt **reduziert**:$\Longleftrightarrow$ entweder $P = \phi$ oder jedes $A \in N$ ist produktiv und erreichbar.

Lemma: Jedes $G \in CFG(\Sigma)$ läßt sich in eine äquivalente reduzierte G, $G' \in CFG(\Sigma)$, transformieren.
Beweis: Stelle für jedes $A \in N$ mit Hilfe von *NFA*'s ob A erreichbar und produktiv.
Fall 1: S nicht produktiv. Wähle $P = \phi$
Fall 2: S produktiv. Weglasse aller $A \in N$, die nicht produktiv oder nicht erreichbar sind, sowie aller Regeln, in denen solche A's vorkommen.

Korollar: Das ϕ-Problem von *CFG*'s ist entscheidbar.

Elimination von ε-Regeln

Definition: $G \in CFG(\Sigma)$ heißt ε-**frei**: $A \longrightarrow \varepsilon \in P \Longleftrightarrow A = S$ und S auf keiner rechten Regelseite.

Lemma: $\chi \overset{*}{\curvearrowright} \varepsilon \Longleftrightarrow A \in \underline{\mathsf{Pre}}^*(\{\varepsilon\})$

Satz: Jedes $G \in CFG(\Sigma)$ läßt sich in eine äquivalente ε-freie Grammatik $G' \in CFG(\Sigma)$ transformieren.
Beweis: Konstruiere mit $\underline{\mathsf{Pre}}^*(\{\varepsilon\})$ die Menge $N_\varepsilon := \{A \in N | N \overset{*}{\curvearrowright}\}$ und damit $G' =< N', \Sigma, P', S' >\in CFG$:

- $N' := N \dot{\cup} \{S'\}$
- $P' := \{S' \longrightarrow S\} \cup \{S' \longrightarrow \varepsilon | S \in N_\varepsilon\} \cup$
  $\{A \longrightarrow X_1 \ldots X_k | k \geq 1, X_i \in (N \cup \Sigma) \exists \alpha_0, \alpha_1, \ldots, \alpha_k \in N_\varepsilon^* : A \longrightarrow \alpha_0 X_1 \alpha_1 \ldots X_k \alpha_k \in P\}$

Beachte: Wegen $k \geq 1$ entfallen ε-Regeln und G' ist ε-frei.
Bleibt zu zeigen: $L(G') = L(G)$

1. $w = \varepsilon$: $\varepsilon \in L(G') \Longleftrightarrow S \in N_\varepsilon \Longleftrightarrow S \curvearrowright_G \varepsilon \Longleftrightarrow \varepsilon \in L(G)$
2. $w \neq \varepsilon$
   (a) $w \in L(G')$ d.h.
   $\Longrightarrow S \overset{*}{\curvearrowright}_{G'} w \Longrightarrow S \overset{*}{\curvearrowright}_G w \Longrightarrow w \in L(G)$ weil Ableitung in G' mit $\alpha_i \overset{*}{\curvearrowright}$ aufgefüllt werden kann zu Ableitung in G.

(b) $w \in L(G)$
Wir zeigen durch Induktion über Ableitungslänge r, daß gilt: $A \underset{r}{\curvearrowright_G} w \in \Sigma^+ \implies A \overset{*}{\curvearrowright_{G'}} w$
$r = 1$: $A \curvearrowright w, w \neq \varepsilon \implies A \longrightarrow w \in P \implies A \longrightarrow w \in P' \implies A \overset{*}{\curvearrowright_{G'}} w$
$r \to r+1$: $A \curvearrowright X_1 \dots X_k \underset{r}{\curvearrowright_G} w$
Dann existieren Ableitungen $X_i \underset{r}{\curvearrowright_G} w_i$ mit $w = w_1 \dots w_k$, $r_i \leq r$
Falls $w_i \neq \varepsilon$, $X_i \overset{*}{\curvearrowright_{G'}} w_i$ nach Induktionsvoraussetzung.
Falls $w_i = \varepsilon$, so $X_i \in N_\varepsilon$. Durch entsprechendes Löschen entsteht $A \curvearrowright_{G'} X'_1 \dots X'_j \overset{*}{\curvearrowright} w'_1 \dots w'_j = w$.

Elimination von Kettenregeln

Definition: Eine Regel der Form $A \longrightarrow B$ heißt **Kettenregel**.

Satz: Jedes $G =< N, \Sigma, P, S > \in$ *CFG* läßt sich in eine äquivalente ε-freie Grammatik $G' =< N', \Sigma, P', S > \in$ *CFG* ohne Kettenregel transformieren:
Beweis: O.B.d.A. sei G ε-frei. Wir definieren $G' =< N, \Sigma, P', S >$ durch $A \longrightarrow \alpha \in P' :\iff \alpha \notin N$ und es gibt $B \in N$ mit $A \in \underline{\mathrm{Pre}}^*(B)$ und $B \longrightarrow \alpha \in P$.

Die Chomsky-Normalform

Definition: $G =< N, \Sigma, P, S > \in$ *CFG* ist in **Chomsky-Normalform** ($G \in$ *CNF*) $:\iff$ Jede Regel von P hat die Form:

- $A \longrightarrow BC$ mit $B, C \in N$ oder
- $A \longrightarrow a$ mit $a \in \Sigma$ oder
- $S \longrightarrow \varepsilon$ und S auf keiner rechten Regelseite.

Satz: Jedes $G =< N, \Sigma, P, S > \in$ *CFG* läßt sich in eine äquivalente Grammatik $G' \in$ *CNF* transformieren.
Beweis: O.B.d.A. sei G ε-frei und ohne Kettenregeln. Außerdem können wir annehmen, daß für $k \geq 2$ $A \longrightarrow X_1 \dots X_k \in P \implies X_i \in N$. Denn $X_i = a$ kann ersetzt werden durch neues $C_i \in N$ unter hinzufügen von $C_i \longrightarrow a$.

Die Greibach Normalform, Linksrekursion

Definition: $G =< N, \Sigma, P, S > \in$ *CFG* ist in Greibach-Normalform ($G \in$ *GNF*) $:\iff$ Jede Regel von P hat die Form

- $A \longrightarrow aB_1 \dots B_n$ oder
- $A \longrightarrow a$ oder
- $S \longrightarrow \varepsilon$ und S auf keiner rechten Regelseite.

Bemerkung: G rechtslinear ($A \longrightarrow wB,\ A \longrightarrow w$) läßt sich offensichtlich *GNF* äquivalent transformieren.

Satz: Jedes $G \in CFG$ läßt sich in äquivalente $G' \in GNF$ transformieren.
Beweisidee: Elimination linksrekursiver N-Symbole.

Definition: $A \in N$ **linksrekursiv** $:\Longleftrightarrow\ A \overset{*}{\curvearrowright} A\alpha$ für ein $\alpha \in \chi^*$
Spezialfall: Direkte Linksrekursion:
Seien $A \longrightarrow A\alpha_1|A\alpha_2|\ldots A\alpha_r|$ alle Regeln der Form $A \longrightarrow A\alpha$
$A \longrightarrow \beta_1|\ldots|\beta_s$
Transformation:

neuer Regelsatz: $A \longrightarrow \beta_1 Z|\ldots|\beta_s Z|\beta_1|\beta_s$
$Z \longrightarrow \alpha_1 Z|\ldots|\alpha_r Z|\alpha_1|\ldots|\alpha_r$

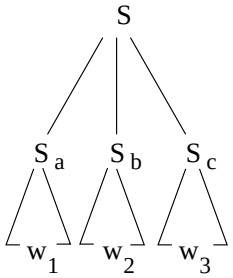
3.4 Abschlußeigenschaften von CFL, Pumping-Lemma

Hilfsmittel: Substitutionssatz, Pumping-Lemma.

Definition: Sei Σ ein Alphabet und für jedes $a \in \Sigma$ sei Σ_a ein weiteres Alphabet.
Dann heißt $\varphi : \mathfrak{P}(\Sigma^*) \longrightarrow \mathfrak{P}\left((\bigcup_{a\in\Sigma} \Sigma_a)^*\right)$ eine **Substitutionsabbildung**, falls

- $\varphi(\{a\}) \subseteq \Sigma_a^*$
- $\varphi(\{\varepsilon\}) = \{\varepsilon\}$
- $\varphi(\{a_1 \ldots a_r\}) = \varphi(\{a_1\}) \cdot \varphi(\{a_2\}) \cdot \cdots \cdot \varphi(\{a_r\})$
- $\varphi(L) = \bigcup_{w\in L} \varphi(\{w\})$

Substitutionssatz:
Sei $\varphi : \mathfrak{P}(\Sigma^*) \longrightarrow \mathfrak{P}\left((\bigcup_{a\in\Sigma} \Sigma_a)^*\right)$ eine Substitutionsabbildung. Dann gilt:
$L \in CFL(\Sigma), \varphi(a) \in CFL(\Sigma_a)$ für alle $a \in \Sigma \implies \varphi(L) \in CFL(\bigcup_{a\in\Sigma} \Sigma_a)$
Beweis: Kombination Kontextfreier Grammatiken mit disjunkten N-Symbolmengen. Ersetze $a \in \Sigma$ durch S_a:



Korollar: $CFL(\Sigma)$ ist unter regulären Operationen abgeschlossen.
Beweis: Seien $L_1, L_2 \in CFL(\Sigma)$ und a, b Buchstaben mit $\varphi(a) = L_1$ und $\varphi(b) = L_2$. Dann gilt:

$$\varphi(\{a,b\}) = L_1 \cup L_2$$
$$\varphi(ab) = L_1 \cdot L_2$$
$$\varphi(\{a\}^*) = L_1^*$$

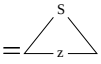
Da $\{a,b\}$, $\{ab\}$, $\{a\} \in CFL(\{a,b\})$, folgt die Behauptung.

Für den fehlenden Abschluß unter $\cap$ und $^-$ betrachten wir das Pumping Lemma ($uvwxy$-Theorem).

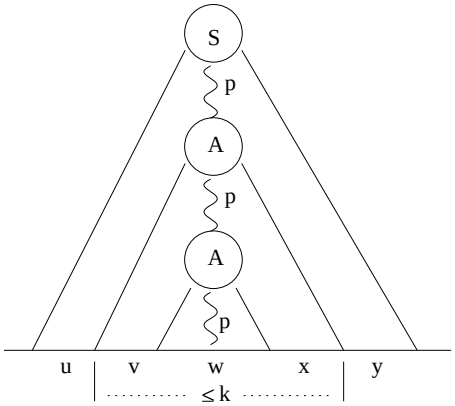
Pumping-Lemma

Sei $L \subseteq CFL(\Sigma)$.
Dann existiert $k \geq 1$, so daß für alle $z \in L$ mit $|z| \geq k$ gilt:
es existiert eine Zerlegung $z = uvwxy$ mit

- $|vwx| \leq k$,
- $vx \neq \varepsilon$ und
- $uv^iwx^iy \in L$ für alle $i \in \mathbb{N}$.

Beweis: O.B.d.A. sei $G =< N, \Sigma, P, S > \in CNF$ mit $L(G) = L \backslash \{\varepsilon\}$.
Sei $n := |N|$, $k := 2^n$ und $z \in L$ mit $|z| \geq k$.
Ein Ableitungsbaum $t =$ 

hat mindestens 2^n Blätter. Da $G \in CNF$, muß ein Pfad p maximaler Länge mindestens $n+1$ Kanten, also $n+2$ Knoten besitzen. p besitzt daher mindestens $n+1$ Knoten mit N-Symbolen. Also gibt es 2 Knoten mit gleichem N-Symbol.



Wähle auf p den letzten Knoten, dessen Marke $A \in N$ sich wiederholt. Dann hat sein Teilbaum höchstens $2^n = k$ Blätter.

- $|vwx| \leq 2^n = k$
- $vx \neq \varepsilon$ weil $G \in CNF$ $(A \longrightarrow BC)$
- $uv^iwx^iy \in L$ für alle $i \in \mathbb{N}$

Lemma: $L = \{a^n b^n c^n | n \geq 1\} \notin CFL(\{a, b, c\})$
Beweis: Währe L kontextfrei, so existiert ein Pumping-Index $k \in \mathbb{N}$. Für $a^k b^k c^k$ existiert Zerlegung $uvwxy$ mit den Eigenschaften des Pumping-Lemmas. Insbesondere $uwy \in L$
$a \ldots ab \ldots bc \ldots c = \underbrace{uvwxy}_{\leq k}$
Da $vx \neq \varepsilon$, muß $|uwy| < 3k$ sein. Da $|vwx| \leq k$, kann vx nicht gleichzeitig ein a und ein c enthalten. Also muß $uwy = \ldots c^k$ sein. Widerspruch.

Satz: *CFL* ist weder unter Durchschnitt noch unter Komplement abgeschlossen.
Beweis:
$S \longrightarrow Sc \,|\, Ac$, $A \longrightarrow aAb \,|\, ab$ und $S \longrightarrow aS \,|\, aA$, $A \longrightarrow bAc \,|\, bc$
erzeugen die Sprachen $\{a^n b^n c^k \,|\, n, k \geq 1\}$ beziehungsweise $\{a^k b^n c^n \,|\, k, n \geq 1\}$ deren Schnitt nicht Kontextfrei ist.
Da *CFL* unter Vereinigung abgeschlossen ist, kann *CFL* nicht unter Komplement abgeschlossen sein, denn $L_1 \cap L_2 = \overline{\bar{L_1} \cup \bar{L_2}}$

3.5 Entscheidbare Eigenschaften

Satz: Das Wortproblem und das Leerheitsproblem sind für *CFG* entscheidbar.
Beweis:
$w \in L(G) \iff S \in \underline{\mathsf{Pre}}^*(\{w\})$
$L(G) = \phi \iff S \notin \underline{\mathsf{Pre}}^*(\Sigma^*)$

Bemerkung:

1. Das Wortproblem ist in $O(n^3)$ entscheidbar.
2. Das Äquivalenzproblem ist nicht entscheidbar.

3.6 Kellerautomaten

- Kellerspeicher (Stack, Stapel): wichtige Datenstruktur für die sequentielle Behandlung strukturierter Objekte.
- Charakterisierung von *CFL* durch nicht-deterministische Kellerautomaten. (Keine Äquivalenz zu deterministischen Kellerautomaten.)
- Bedeutung deterministischer Kellerautomaten für den Compilerbau
  - Syntaxanalyse
  - Datenkeller (Auswertung von Rechner ausdrücken)
  - Prozedurkeller (Speichertechnik für rekursive Prozeduren)

Definition: Seien Q, Σ, Γ nicht-leere, endliche Mengen von Zuständen, Eingabe- und Kellersymbolen, ferner $q_0 \in Q$ Anfangszustand, $F \subseteq Q$ Endzustandsmenge,$z_0 \in \Gamma$ Kellerstartsymbol und $\delta : Q \times \Sigma_\varepsilon \times \Gamma \longrightarrow Pot_f(Q \times \Gamma^*)$ Transitionsfunktion[1].
Dann heißt $\mathfrak{A} =< Q, \Sigma, \Gamma.\delta, q_0, F, z_0 >$ ein **Kellerautomat** über Σ.
Bezeichnung: $\mathfrak{A} \in PDA(\Sigma)$

[1] Das f in Pot_f bedeutet: endliche Teilmenge

Semantik:

Konfigurationsmenge: $Q \times \Sigma^* \times \Gamma^*$ wobei die Kellerspitze links sei.

Einzelschrittrelation:
$(q, w, \alpha) \vdash (q', w', \alpha')$

- Σ-Schritt
  $(q, aw, Za) \vdash (q', w, \beta\alpha)$, falls $\delta(q, a, Z) \ni (q', \beta)$
- ε-Schritt
  $(q, w, Z\alpha) \vdash (q', w, \beta\alpha)$, falls $\delta(q, \varepsilon, z) \ni (q', \beta)$

Die von $\mathfrak{A}$ erkannte Sprache

$L(\mathfrak{A}) := \{w \in \Sigma^* \mid (q_0, w, Z_0) \overset{*}{\vdash} (q, \varepsilon, \alpha),\ q \in F\}$

Bemerkung: alternative Erkennung mit leerem Keller äquivalent
$\mathfrak{L}(PDA, F) = \mathfrak{L}(PDA, \varepsilon)$

Satz: $CFL(\Sigma) \subseteq \mathfrak{L}(\Sigma, PDA)$
Beweis: Simulation von links-Ableitungen auf Keller.
Sei $G =< N, \Sigma, P, S > \in CFG$.
Definiere $\mathfrak{A}_G =< G, \Sigma, \Gamma, \delta, q_0, F, Z_0 > \in PDA(\Sigma)$ durch $Q := \{q\}$, $\Gamma : N \cup \Sigma$, $Z_0 := S$,
$\delta(q, a, a) := \{(q, \varepsilon)\}$ für alle $a \in \Sigma$
$\delta(q, \varepsilon, A) := \{(q, \beta) | A \longrightarrow \beta \text{ in } P\}$ für alle $A \in N$
Ziel: Nachweis von $L(G) = L(\mathfrak{A}_G, \varepsilon)$
(*) $A \overset{*}{\curvearrowright} w \iff (q, wv, A\alpha) \overset{*}{\vdash} (q, v, \alpha)$ für alle $v \in \Sigma^*$, $\alpha \in \Gamma^*$.
"$\Longrightarrow$": Induktion über Ableitunslänge n:
$n = 1$: $A \longrightarrow w$, $(q, wv, A\alpha) \vdash (q, wv, w\alpha) \overset{*}{\vdash} (q, v, a)$
$n \mapsto n + 1$: $A \curvearrowright v_0 A_1 v_1 \ldots A_r v_r \overset{n}{\curvearrowright} w$
Dann muß $A_i \overset{n_i}{\curvearrowright} w_i$, $w = v_0 w_1 v_1 \ldots w_r v_r$, so daß nach Induktion:
$(q, \underbrace{v_0 w_1 \ldots w_r v_r}_{w} v, A\alpha) \vdash (q, v_0 w_1 \ldots v_r v, v_0 A_1 v_1 \ldots A_r v_r \alpha)$
$\overset{*}{\vdash} (q, w_1 v_1 \ldots v_r v, A_1 v_1 \ldots A_r v_r \alpha)$ nach Induktions Voraussetzung:
$\overset{*}{\vdash} (q, v_1 w_2 \ldots v_r v, v_1 \ldots \alpha) \overset{*}{\vdash} \ldots \overset{*}{\vdash} (q, v, \alpha)$

"$\Longleftarrow$": (Induktion über die Länge der Berechnung n)
$n = 1$: Dann muß $A \longrightarrow \varepsilon$ und $w = \varepsilon$ gelten, also $A \overset{*}{\curvearrowright} w = \varepsilon$
$n \mapsto n + 1$: $(q, wv, A\alpha) \overset{n+1}{\vdash} (q, v, a)$ zerfällt in
$(q, wv, A\alpha) \vdash (q, wv, v_0 A_1 v_1 \ldots A_r v_r \alpha)$
$\overset{*}{\vdash} (q, w_1 v_1 \ldots w_r v_r v, A_1 v_1 \ldots A_r v_r \alpha)$
$\overset{n_1}{\vdash} (q, w_1 v_1 \ldots w_r v_r v, v_1 \ldots A_r v_r \alpha)$
$\vdots$
$\overset{n_r}{\vdash} (q, v_r v, v_r \alpha)$
$\overset{*}{\vdash} (q, v, \alpha)$
Nach Induktionsvorraussetzung gibt es Ableitungen
$A_1 \overset{*}{\curvearrowright} w_1$

$\vdots$
$A_r \overset{*}{\curvearrowright} w_r$, so daß $A \curvearrowright v_0 A_1 v_1 \dots A_r v_r \overset{*}{\curvearrowright} v_0 w_1 \dots v_r = w$

Satz: $\mathfrak{L}(\Sigma, PDA, F) \subseteq CFL(\Sigma)$ (ohne Beweis)

Korollar: $\mathfrak{L}(\Sigma, PDA, F) = CFL(\Sigma)$

Korollar: *CFL* ist unter Schnitt mit regulären Sprachen abgeschlossen.
Beweis: Für $L \in CFL(\Sigma)$ und $R \in REG(\Sigma)$ existiert $\mathfrak{A}_L \in PDA(\Sigma)$ und $\mathfrak{A}_R \in DFA(\Sigma)$ mit $L = L(\mathfrak{A}_L)$ und $R = L(\mathfrak{A}_R)$.
Konstruiere $\mathfrak{A}_L || \mathfrak{A}_R \in PDA(\Sigma)$ durch Parallel-Konstruktion.
$Q := Q_L \times Q_R$ und
$\delta((q_1, q_2), a, z) \ni ((q_1', q_2'), \alpha)$ falls $\delta_L(q_1, a, z) \ni (q_1', \alpha)$ und $\delta_R(q_2, a) = q_2'$
$F := F_L \times F_R$

Ziel: Der Deklarationszwang von Variablen ist keine kontextfreie Eigenschaft.

Lemma: $L = \{ww | w \in \{a,b\}^+\} \notin CFL$
Beweis: Zunächst zeigen wir $L' = \{a^m b^n a^m b^n \mid m, n \geq 1\} \notin CFL$.
Wäre $L' \in CFL$, so existiert Pumping-Index k mit $a^k b^k a^k b^k = uvwxy$ mit $|vwx| \leq k$ und $vx \neq \varepsilon$, also auch $uwy \in L'$. Das ist ein Wiederspruch, weil die Zahl der vorderen und hinteren a's beziehungsweise b's nicht mehr gleich ist.
$L' = L \cap \{a\}^+\{b\}^+\{a\}^+\{b\}^+$
Wäre $L \in CFL$, so auch L'. Wiederspruch.

Korollar: Die Menge P der Pascal-Programme ist nicht kontextfrei.

Beweis: $L := \{$program $T(output)$; var $w : integer$;
begin $w :=$ 1 end. $\mid w \in \{a,b\}^+\}$
Also $L \subseteq P$
R sei bestimmt durch den regulären Ausdruck
program $T(output)$; var$(a \vee b)^+ : integer$;
begin$(a \vee b)^+ :=$ 1 end.
Dann gilt: $L = P \cap R$
h sei eine subst. Abbildung mit $h(a) = a$ $h(b) = b$ $h(x) = \varepsilon$.
Dann $h(L) = \{ww \,| w \in \{a,b\}^+\} \notin CFL \implies L \notin CFL \implies P \notin CFL$.

Deterministische Kellerautomaten

Ziel: $\mathfrak{L}(\Sigma, DPDA)$ unter Komplement abgeschlossen, somit $\mathfrak{L}(\Sigma, DPDA) \subsetneq \mathfrak{L}(\Sigma, PDA) = CFL$, das heißt im allgemeinen keine Erkennung von $L \in CFL$ durch deterministische Kellerautomaten möglich.
Syntaxanalyse mit $DPDA's$ für spezielle *CFG*.

Definition: Sei $\mathfrak{A} =< Q, \Sigma, \Gamma, \delta, q_0, Z_0, F > \in PDA$
$\mathfrak{A}$ heißt **deterministisch**, in Zeichen $\mathfrak{A} \in DPDA(\Sigma)$, wenn gilt:

1. $|\delta(q, a, Z)| \leq 1$ für alle $(q, a, Z) \in Q \times \Sigma_\varepsilon \times \Gamma$
2. $|\delta(q, \varepsilon, Z)| = 1 \implies |\delta(q, a, Z)| = 0$ für alle $a \in \Sigma$

Folgerung: Zu jeder Konfiguration (q, w, α) gibt es höchstens eine Folgekonfiguration.ε-Schritte möglich.

Bemerkung: Bei Erkennung mit leerem Keller sind weniger Sprachen erkennbar.

Satz: $\mathfrak{L}(\Sigma, DPDA)$ ist unter Komplement abgeschlossen.
Beweisidee: Zu jedem $\mathfrak{A} \in DPDA(\Sigma)$ ist ein äquivalenter $\tilde{\mathfrak{A}} \in DPDA(\Sigma)$ konstruierbar, so daß gilt:
Für jedes $w \in \Sigma^*$ gibt es $\tilde{Q} \in Q$ und $\alpha \in \tilde{\Gamma}^*$ mit $(\tilde{q}_0, w, \tilde{Z}_0) \vdash^* (\tilde{q}, \varepsilon, \alpha)$.
Dazu müssen Schleifenkonstruktionen eliminiert werden. $(q, w, \alpha) \in Q \times \Sigma^* \times \Gamma^*$ heißt **Schleifenkonfiguration**, falls für jedes $i \in \mathbb{N}$ ein $q_i \in Q$ und ein $\alpha_i \in \Gamma^*$ existiert, so daß $|\alpha_i| \geq |\alpha|$ und $(q, w, \alpha) \vdash^i (q_i, w, \alpha_i)$
Diese Eigenschaft ist entscheidbar $\Longrightarrow$ Elimination.

3.7 Der Algorithmus von Cocke, Younger und Kasami

Effiziente Lösung des Wortproblems für beliebige *CFL*. Dynamische Programmierung, $O(n^3)$-Zeit, $O(n^2)$-Platz. O.B.d.A. $G \in CNF$ (auf beliebige *CFG* übertragbar).
Für $G =< N, \Sigma, P, S > \in CNF$ (* $A \longrightarrow BC$, $A \longrightarrow a$ *) und $w = a_2 a_2 \ldots a_n$ mit $n > 0$ definieren wir
$w_{ij} := a_i a_{i+1} \ldots a_j$ für $1 \leq i \leq j \leq n$
$N_{ij} := \{A \in N | A \curvearrowright^* w_{ij}\}$
so daß gilt: $w \in L(G) \iff S \in N_{1n}$

Bestimmen von N_{1n}:

Bestimme	$N_{1\,1}\, N_{2\,2}\, N_{3\,3} \ldots N_{n\,n}$
dann	$N_{1\,2}\, N_{2\,3} \ldots N_{n-1\,n}$
	$N_{1\,3}\, N_{2\,4} \ldots N_{n-2\,n}$
	$\vdots$
bis	N_{1n}

mit Hilfe von

1. $A \in N_{i\,i} \iff A \longrightarrow a_i \in P$
2. $A \in N_{i\,j}$ mit $i < j \iff \exists k : i \leq k < j, \exists A \longrightarrow BC \in P\ B \in N_{i\,k},\ C \in N_{k+1\,j}$

Beispiel: Sei G gegeben durch folgende Regeln:
$S \longrightarrow AB \mid a$
$A \longrightarrow BA \mid a$
$B \longrightarrow AB \mid b$
Dann ergibt sich nach dem Algorithmus folgende Tabelle:

i \ j	1	2	3	4
1	$\{B\}$	$\{A\}$	ϕ	$\{B, S\}$
2		$\{S, A\}$	ϕ	$\{S, B\}$
3			$\{S, A\}$	$\{S, B\}$
4				$\{B\}$
	b	a	a	b

Komplexität:
a) Zeit:

äußere	"`For i`",	$c_1 \cdot n$	Schritte
	"`For k`",	$c_2 \cdot d$	Schritte
äußere	"`For i`",	$c_2 \cdot (n-d) \cdot n$	Schritte
	"`For d`",	$c_2 \sum_{d=1}^{n-1} (n-d) \cdot d$	Schritte

Insgesamt $c \sum_{d=1}^{n} (n-d)d = c(n \sum_{d=1}^{n} d - \sum_{d=1}^{n} d^2)$
$= c(n^2 \frac{n+1}{2} - n \frac{(n+1)(2n+2)}{6})$
$= c \frac{n^3 - n}{6} \quad = O(n^3)$

b)Platz: $O(n^2)$

3.8 Erweiterte Kontextfreie Grammatiken (*EBNF*)

Höherer Beschreibungskomfort durch reguläre Ausdrücke als rechte Regelseite; äquivalente Erweiterung.

Definition: $G =< N, \Sigma, P, S >$ ist eine **erweiterte** *CFG*, in Zeichen $G \in$ *ECFG*, wenn $< N, \Sigma, \phi, S > \in$ *CFG* und $P : N \longrightarrow RA(N \cup \Sigma)$
Schreibweise: $A \longrightarrow \alpha$, falls $P(A) = \alpha$

Semantik: P repräsentiert die Regelmenge $\tilde{P}$ mit
$A \longrightarrow \tilde{\alpha} \in \tilde{P} :\Longleftrightarrow \tilde{\alpha} \in [\![P(A)]\!]$
Beachte: $\tilde{P}$ ist im allgemeinen unendlich $L(G) := L(< N, \Sigma, \tilde{P}, S >)$

Satz: $CFL(\Sigma) = \mathfrak{L}(\Sigma, ECFG)$
Beweisidee: "$\subseteq$": nach Definition
"$\supseteq$": Transformation von *ECFG* nach *CFG* durch Elimination regulärer Ausrücke.

- Disjunktion: Regelalteration
- Stern: neues Nichtterminalsymbol mit $A \longrightarrow \alpha A \mid \varepsilon$

3.9 Rekursiv endliche Automaten (Syntaxdiagramme)

Syntaxdiagramme entsprechen endlichen Automaten, die sich rekursiv aufrufen können. Neben $\underset{q}{\bullet} \xrightarrow{a} \underset{q'}{\bullet}$ auch $\underset{q}{\bullet} \xrightarrow{[r]} \underset{q'}{\bullet}$ möglich.

Definition: $\mathfrak{A} =< Q, \Sigma, \delta, q_0, F >$ heißt **rekursiv endlicher Automat** über Σ, falls $\delta : Q \times (\Sigma_\varepsilon \cup Q) \longrightarrow \mathfrak{P}(Q)$ und sonstwie ein endlicher Automat. Bezeichnung: $\mathfrak{A} \in RFA(\Sigma)$

Semantik:
Konfiguration: $Q \times \Sigma^*$
Transitionsrelation: $\vdash \subseteq (Q \times \Sigma^*)^2$ definiert durch [2] $\frac{p \in \delta(q,a)}{(q,aw) \vdash (p,w)}$ $\frac{p \in \delta(q,\varepsilon)}{(q,w) \vdash (p,w)}$ (wie bei *NFA*'s)
$\frac{p \in \delta(q,r)\ (r,w) \vdash^* (s,v)\ s \in F}{(q,w) \vdash (p,v)}$
$L(\mathfrak{A}) := \{w \in \Sigma^* \mid (q_0, w) \vdash^* (p, \varepsilon),\ p \in F\}$

Satz: $\mathfrak{L}(\Sigma, RFA) = \mathfrak{L}(\Sigma, PDA)$
Beweis: "$\subseteq$": $\mathfrak{A} =< Q, \Sigma, \delta, q_0, F > \in RFA(\Sigma)$
Simulation von $\mathfrak{A}$ durch $\mathfrak{A}' =< Q, \Sigma, Q, \delta', q_0, q_0, \phi > \in PDA$
Idee: Keller für "Rücksprungadressen" (Zustände)$\in PDA$
$\delta'(q, a, s) = \{(p, s) | p \in \delta(q, a)\}$

$$\begin{array}{lll} \delta'(q, \varepsilon, s) = & \{(p, s) | p \in \delta(q, \varepsilon)\} & \\ & \cup\{(r, p, s) | p \in \delta(q, r)\} & \text{"Aufruf"} \\ & \cup\{(s, \varepsilon) | q \in F\} & \text{"Rücksprung"} \end{array}$$

für alle $q, s, r, p \in Q$ und $q \in \Sigma$
"$\supseteq$": Für $L \in \mathfrak{L}(\Sigma, PDA)$ existiert $G \in < N, \Sigma, P, S > \in CNF$ mit $L = L(G)$. Konstruiere $\mathfrak{A} \in <$

[2] Notation wie in der Mathematischen Logik. Oben stehen die Bedingungen, unten die Folgerung.

$Q, \Sigma, \delta, q_0, F > \in RFA$

durch	$Q := N \dot{\cup} q_f,\ q_0 := S,\ F := \{q_f\}$	
und	$C \in \delta(A, B)$	falls $A \longrightarrow BC$
	$q_f \in \delta(A, a)$	falls $A \longrightarrow a$
	$q_f \in \delta(S, \varepsilon)$	falls $S \longrightarrow \varepsilon$

Kapitel 4

Turingmaschinen und aufzählbare Sprachen

4.1 Chomsky-Grammatiken

(siehe Folie F4.1)

Verallgemeinerung kontextfreier Grammatiken. Kontextabhängige Ersetzung, Wortersetzung.

Definition: Seien N, Σ, χ und S wie bei *CFG*. Sei ferner $P \subseteq \chi^* N \chi^* \times \chi^*$, P endlich. Dann heißt $G =< N, \Sigma, P, S >$ eine **Chomsky-Grammatik**
Semantik: $\pi = \alpha_1 \longrightarrow \alpha_2 \in P$ bestimmt die **Ableitungsrelation** $\curvearrowright_\pi \subseteq \chi^* \times \chi^*$ durch $\beta_1 \curvearrowright_\pi \beta_2 :\Longleftrightarrow$ es existieren $\gamma, \delta \in \chi$ mit $\beta_1 = \gamma\alpha_1\delta$ und $\beta_2 = \gamma\alpha_2\delta$
Ablteitungsrelation von G: $\curvearrowright_G := \bigcup_{\pi \in P} \curvearrowright_\pi$
G erzeugt $L(G) := \{w \in \Sigma^* | S \overset{*}{\curvearrowright}_G w\}$

(siehe Folie F4.2)

Abschlußeigenschaften von $\mathfrak{L}_0$ und $\mathfrak{L}_1$

Definition: Sei $G =< N, \Sigma, P, S >$ von Typ 0. G heißt **normiert**, wenn gilt:

- Für jedes $\pi = \alpha_1 \longrightarrow \alpha_2 \in P$ gibt es $\gamma, \delta \in N^*$, $A \in N$, $\beta \in \chi^*$ mit $\alpha_1 = \gamma A \delta$ und $\alpha_2 = \gamma\beta\delta$
- Σ-Symbole nur in Regeln der Form $A \longrightarrow a$

Beachte: $\beta = \varepsilon$ erlaubt im Gegensatz zu Typ 1.

Satz: Zu jedem $G =< N, \Sigma, P, S >$ vom Typ 0 läßt sich eine äquivalente normierte Grammatik $G' =< N', \Sigma, P', S' >$ konstruieren.
Beweis: 1.) Terminalsymbol-Bedingung.
$a \in \Sigma \mapsto A_a$ neues Nichtterminal-Symbol.
a durch A_a ersetzen (in P).
$A_a \longrightarrow a$ hinzufügen.
2.) Simulation von $A_1 \ldots A_n \longrightarrow B_1 \ldots B_m$ $(n \geq 1, m \in \ltimes)$ durch folgende Regeln:
$A_1 \ldots A_n \longrightarrow A'_1 A_2 \ldots A_n$

$A'_1 A_2 \dots A_n \longrightarrow A'_1 A'_2 A_3 \dots A_n$
$\vdots$
$A'_1 \dots A'_{n-1} A_n \longrightarrow A'_1 \dots A'_n$
$A'_1 \dots A'_n \longrightarrow B_1 A'_2 \dots A'_n$
$\vdots$
2 Fälle:
1. Fall: $n \leq m \quad B_1 \dots B_{n-1} A'_n \longrightarrow B_1 \dots B_n$
2. Fall: $n > m \quad B_1 \dots B_m A'_{m+1} \dots A'_n \longrightarrow B_1 \dots B_m A'_{m+2} \dots A'_n \ \dots \longrightarrow B_1 \dots B_m$
A'_n neue Nichtterminalsymbole Verhindern Seiteneffekte (neue Satzformen, mehr Ableitungen).

Satz: $\mathfrak{L}_0(\Sigma)$ ist unter Substitution abgeschlossen, das heißt: ist $\varphi : \mathfrak{P}(\Sigma^*) \longrightarrow \mathfrak{P}\left(\left(\bigcup_{a\in\Sigma} \Sigma_a\right)^*\right)$ eine Substitutions Abbildung, so gilt:
$L \in \mathfrak{L}_0(\Sigma)$, $\varphi(a) \in \mathfrak{L}_0(\Sigma_a)$ für alle $a \in \Sigma$
$\Longrightarrow \varphi(L) \in \mathfrak{L}_0(\bigcup_{a\in\Sigma} \Sigma_a)$
Beweis: Konstruktion von *CFG* nicht anwendbar wegen möglicher neuer Satzformen.
Idee: Sequentialisierung mit einem Kontrollsymbol. $L = L(G)$ mit $G =< N, \Sigma, P, S >$ vom Typ 0.
$\varphi(A) = L_a = L(G_a)$ mit $G_a =< N_a, \Sigma_a, P_a, S_a >$ vom Typ 0.
O.B.d.A.: G, G_a normiert, N, N_a disjunkt.
Konstruktion von $\bar{G}$ mit $L(\bar{G})$ mit $L(\bar{G}) = \varphi(L)$ $\bar{N} := N \dot\cup (\dot\bigcup_{a\in\Sigma} N_a) \dot\cup \{A_a | a \in \Sigma\} \dot\cup \{\bar{S}, \underset{\uparrow!}{C}\}$
$\bar{Sigma} = \bigcup_{a\in\Sigma} \Sigma_a$
$\bar{P} := P_0 \cup P_1 \cup P_2 \cup (\bigcup_{a\in\Sigma} P_a) \cup \{\bar{S} \longrightarrow CS,\ C \longrightarrow \varepsilon\}$
P_0 entstehe aus P durch Ersetzen von a durch A_a $(a \in \Sigma)$
$P_1 := \{CA_a \longrightarrow CS_a \,|\, a \in \Sigma\}$
$P_2 := \{C_a \longrightarrow aC \,|\, a \in \bar{\Sigma}$

1. $\varphi(L) \subseteq L(\bar{G})$L
   $\bar{S} \curvearrowright CS \overset{*}{\curvearrowright} CA_{a_1} \dots A_{a_r}$
   $\overset{*}{\curvearrowright} Cw_1 A_{a_2} \dots A_{a_r}$
   $\curvearrowright w_1 C A_{a_2}$
   $\dot{quad}$:
   $w_1 \dots w_r$
2. $L(\bar{G}) \subseteq \varphi(L)$:
   keine Ableitung neuer Wörter, weil G normiert und daher A_a auf keiner linken Regelseite, weil die N-Alphabete disjunkt und weil C die Teilableitungen der G_a sequentialisiert.

Korollar $\mathfrak{L}_0(\Sigma)$ ist unter den regulären Operationen abgeschlossen. $L, L' \in \mathfrak{L}_0(\Sigma) \iff L \cup L'$, LL', $L^* \in \mathfrak{L}_0(\Sigma)$
Beweis: wie für *CFL*.

Satz: $\mathfrak{L}_0(\Sigma)$ ist unter Schnitt abgeschlossen.
Beweis: $L_i = L(G_i)$ und $G_i =< N_i, \Sigma, P_i, S_i >$ vom Typ 0 für $i = 1, 2$
O.B.d.A. sei $N_1 \cap N_2 = \phi$.
Konstruktion von $G =< N, \Sigma, P, S >$
$N := N_1 \dot\cup N_2 \cup \{A_a | a \in \Sigma\} \dot\cup \{S, C_1, C_2\}$
$P := P_1 \cup P_2 \cup Q$
$A := \{S \longrightarrow C_1 S_2 C_2 S_2 C_1,\ C_2 a \longrightarrow A_a C_2,\ bA_a \longrightarrow A_a b,\ C_1 A_a a \longrightarrow aC_1,\ C_1 C_2 C_1 \longrightarrow \varepsilon\}$

Kontextfreie Grammatiken und Sprachen

Typ 1 $\sim$ Platzbedarf von Berechnungen

Definition: $G =< N, \Sigma, P, S >$ heißt **von wachsender Länge**
$:\Longleftrightarrow$ für jede Regel $\alpha \longrightarrow \beta \in P$ gilt $|\alpha| \leq |\beta|$, es sei denn, daß $S \longrightarrow \varepsilon \in P$ und S auf keiner rechten Regelseite.

Korollar: Eine Typ 1-Grammatik ist von wachsender Länger.

Satz: Zu jeder Grammatik von wachsender Länge läßt sich eine äquivalente Typ 1-Grammatik konstruieren.
Beweis: Normierung.

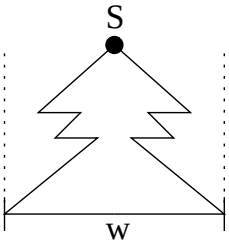
Definition: (Platzbedarf)
Sei $G =< N, \Sigma, P, S >$ von Typ 0, $\delta = (S \curvearrowright \alpha_0 \curvearrowright \alpha_1 \ldots \curvearrowright \alpha_n)$ eine Ableitung von G und $w \in \Sigma^*$
Dann heißt

- $\mathsf{pl}_G(\delta) := \max\{|\alpha_i| \,|\, 0 \leq i \leq n\}$ der **Platzbedarf von** δ und
- $\mathsf{pl}_G(w) := \min\{\mathsf{pl}_G(\delta) \,|\, \delta = (S \curvearrowright \ldots \curvearrowright w)\}$

Folgerung:

1. $\mathsf{pl}_G(w) \geq |w|$
2. G vom Typ 1, $w \neq \varepsilon \iff \mathsf{pl}(w) = |w|$

Platzbedarfssatz:
Sei G vom Typ 0 und $p \in \mathbb{N}$, $p > 0$, so daß für alle $w \in L(G)\backslash\{\varepsilon\}$ gilt: $\mathsf{pl}_g(w) \leq p|w|$.
Dann ist $L(G) \in \mathfrak{L}_1$
Beweis: Sei $p = 1$, also für $w \in L(G)\backslash\{\varepsilon\}$ $\mathsf{pl}_G(w) = |w|$



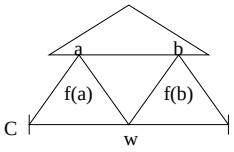
Elimination verkürzender Regeln durch Auffüllen. Konstruktion einer äquivalenten Grammatik von wachsender Länge: $G' =< N \dot\cup \{\$\}, \Sigma, P', S >$

1. $a \longrightarrow \beta \in P$ mit $|\alpha| = |\beta| + i$ $(i > 0)$
   $\Longrightarrow$ $\alpha \longrightarrow \$^i\beta \in P'$
2. $\alpha \longrightarrow \beta \in P$ mit $|\alpha| \leq |\beta|$
   $\Longrightarrow$ $\$^j\alpha \longrightarrow \beta \in P'$ für alle $j = 0, 1, \ldots, |\beta| - |\alpha|$
3. $\$X \longrightarrow X\$$, $X\$ \longrightarrow \in P'$ für alle $X \in \chi$

Wegen $\mathsf{pl}_G(w) = |W|$ lassen sich die eingeschobenen \$ löschen. $p > 1$ Induktion. Idee: $N' \supseteq N^p$.

Abschlußeigenschaften von $\mathfrak{L}_1(\Sigma)$ mit Hilfe des Platzbedarfsatzes

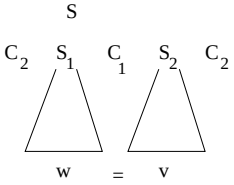
Satz: $\mathfrak{L}_1(\Sigma)$ ist unter ε-freier Substitution $(\varepsilon \notin \varphi(a))$ abgeschlossen.
Beweis:



(Das f in der Zeichnung steht für φ)
Platzbedarf der Ausgangs Grammatik mit $P = 1 \Longleftarrow$ Platzbedarf der Substitutionsgröße $|w|+1 \leq 2|w|$ wegen zusätzlichem Kontrollsymbol C und fehlender ε-Regeln.

Korollar: $\mathfrak{L}_1(\Sigma)$ ist unter regulären Operationen ($\cup$, $\cdot$, *) abgeschlossen.
Beweis: Reduktion auf ε-freie Typ 1-Sprache. $L \in \mathfrak{L}_1(\Sigma) \Longrightarrow L\backslash\{\varepsilon\} \in \mathfrak{L}(\Sigma)$.
Für ε-freie Sprachen $L, L' \in \mathfrak{L}_1(\Sigma)$ folgt $L \cup L'$, LL', $L^* \in \mathfrak{L}_1(\Sigma)$, wie für Typ 0 Sprachen durch ε-freie Substitution. Falls $\varepsilon \in L$ und/oder $\varepsilon \in L'$, so folgert man aus den ε-freien Teilsprachen, zum Beispiel:
$(L \dot\cup \{\varepsilon\})(L' \dot\cup \{\varepsilon\}) = LL' \cup L \cup L' \cup \{\varepsilon\} \in \mathfrak{L}_1(\Sigma)$

Korollar: $\mathfrak{L}_1(\Sigma)$ ist unter Durchschnitt abgeschlossen.
Beweis: $L_1, L_2 \in \mathfrak{L}_1(\Sigma)$ sind mit linearem Platzbedarf ($p = 1$) erzeugbar. Die $\cap$-Konstruktion für Typ 0:



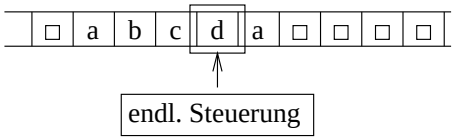
hat dann einen Platzbedarf von $2|w| + 3 \leq 5|w|$

Korollar: $\mathfrak{L}_2(\Sigma) \subsetneq \mathfrak{L}_1(\Sigma)$
Beweis: $\mathfrak{L}_2(\Sigma)$ ist nicht unter $\cap$ abgeschlossen.

Ein langes offenes Problem: Abschluß von $\mathfrak{L}_1$ unter Komplement. 1987: $\mathfrak{L}_1$ ist unter Komplement abgeschlossen.

4.2 Turingmaschinen, linear beschränkte Automaten

Ziel: $\mathfrak{L}_0(\Sigma) = \mathfrak{L}(\Sigma, TM)$
$\mathfrak{L}_1(\Sigma) = \mathfrak{L}(\Sigma, LBA)$



Definition: (Nicht-deterministische erkennende TM)
Seien die folgenden nicht-kleeren, endlichen Mengen gegeben:
Q **Zustandsmenge**
Σ **Eingabealphabet**
Γ **Arbeitsalphabet**
Ferner: $q_0 \in Q$ **Anfangszustand**
$\square \in \Gamma\backslash\Sigma$ **Blank**
$F \subseteq Q$ **Endustandsmenge**
und die Transitionsfunktion:
$\delta : Q \times \Gamma \longrightarrow \mathfrak{P}(Q \times \Gamma \times \{L, N, R\})$

Dann heißt $\mathfrak{A} =< Q, \Sigma, \Gamma, q_0, \Box, F, S >$ eine (nicht-deterministische)**erkennende Turingmaschine**
Bezeichnung: $\mathfrak{A} \in TM(\Sigma)$

Semantik:
Konfigurationsmenge: $Q \times \Gamma^* \times \Gamma \times \Gamma^* =: \mathsf{Conf}(\mathfrak{A})^2$ ist definiert durch

- $(q, \alpha, X, \beta) \vdash (q', \alpha, X', \beta)$ falls $\delta(q, X) \ni (q', X', N)$
- $(q, \alpha Y, X, \beta) \vdash (q', \alpha, Y, X'\beta)$ falls $\delta(q, X) \ni (q', X', L)$
- $(q, \varepsilon, X, \beta) \vdash (q', \varepsilon, \Box, X'\beta)$ falls $\delta(q, X) \ni (q', X', L)$
- $(q, \alpha, X, Y\beta) \vdash (q', \alpha X', Y, \beta)$ falls $\delta(q, X) \ni (q', X', R)$
- $(q, \alpha, X, \varepsilon) \vdash (q', \alpha X', \Box, \varepsilon)$ falls $\delta(q, X) \ni (q', X', R)$

Bemerkung: beiderseits unendliches Band, fast überall $\Box$

Anfangskonfiguration von $w \in \Sigma^*$:
$$K(w) := \begin{cases} (q_0, \varepsilon, a, v) & \text{falls } w = av \\ (q_0, \varepsilon, \Box, \varepsilon) & \text{falls } w = \varepsilon \end{cases}$$

Endkonfiguration (a, α, X', β) falls $q \in F$

4.2.1 Die von $\mathfrak{A}$ erkannte Sprache

$L(\mathfrak{A}) = \{w \in \Sigma^* | K(w) \vdash^* (q, \alpha, X, \beta), q \in F\}$

Beachte: Erreichen von $q \in F$ genügt; kein Anhalten gefordert.

Die Klasse der *TM*-**erkennbaren Sprachen**L $\mathfrak{L}_0(\Sigma, TM)$

Satz: $\mathfrak{L}(\Sigma, TM) = \mathfrak{L}_0(\Sigma)$
Beweis:
I. $\mathfrak{L}(\Sigma, TM) \subseteq \mathfrak{L}_0(\Sigma)$
$\mathfrak{A} =< Q, \Sigma, \Gamma, q_0, \Box, F, \delta > \in TM(\Sigma)$
Simulation von $\mathfrak{A}$ durch Typ 0-Grammatik
Idee: Erzeugen einer beliebigen Anfangskonfiguration mit hinreichend vielen Blanks an den Rändern (bei Ableitungen sind Ränder nicht erkennbar).
Zwei Spuren: 1. Spur hält Eingabe für die Erzeugung
2. Spur simuliert *TM*
Für $a_1 \dots a_2 \in \Sigma^*$, $m, n \in \mathbb{N}$ erzeugt man
(*) $(\varepsilon, \Box)^m q_0(a_1, a_1)(a_2, a_2) \dots (a_r, a_r)(\varepsilon, \Box)^n$
Dazu $G =< N, \Sigma, P, S >$ mit $N := Q \dot\cup (\Sigma_\varepsilon, \Gamma) \dot\cup \{S, C_1, C_2\}$
P: $S \longrightarrow (\varepsilon\Box)S \,|\, q_0C_1$
$C_1 \longrightarrow (a, a)C_1 \,|\, C_2$ $(a \in \Sigma)$
$C - 2 \longrightarrow (\varepsilon, \Box)C_2 \,|\, \varepsilon$
$q(a, X) \longrightarrow q'(a, X')$ falls $\delta(a, X) \ni (q', X', N)$
$q(a, X) \longrightarrow (a, X')q$ falls $\delta(a, X) \ni (q', X', R)$
$(b, Y)q(a, X) \longrightarrow q'(b, Y)(a, X')$ falls $\delta(q, X) \ni (q'.X', L)$
$q \longrightarrow \varepsilon$ falls $q \in F$
$(a, X) \longrightarrow a$ $(a \in \Sigma_\varepsilon)$
Es folgt: $L(G) = L(\mathfrak{A})$
II . $\mathfrak{K}_0(\Sigma) \subseteq \mathfrak{L}(\Sigma, TM)$
Simulation einer Typ-0-Grammatik durch *TM* neben Eingabewort Ableitungen von G simulieren und erzeugtes Wort mit Eingabe vergleichen.
Technisch aufwendig: einschieben und löschen von Wörtern auf einem Turingband.

Automatencharakterisierung von $\mathfrak{L}_1(\Sigma)$ durch Platzbedarf

Definition: (Platzbedarf für *TM*): Sei $\mathfrak{A} \in TM(\Sigma)$, $\gamma = (K_0 \vdash K_1 \vdash K_2 \cdots \vdash K_n)$ eine Berechnung und $w \in L(\mathfrak{A})$. Dann ist:

- $|K_i| = |aX\beta|$ falls $K_i = (q, a, X, \beta)$
- $bv_{\mathfrak{A}}(\gamma) := \max\{|K_i| \,|\, 0 \leq i \leq n\}$
- $bv_{\mathfrak{A}}(w) := \min\{bv_{\mathfrak{A}}(\gamma) \,|\, \gamma \text{ erkennt } w\}$

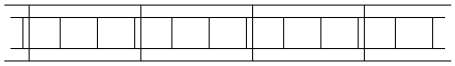
der Bandverbrauch von $\mathfrak{A}$ für γ bzw. w

Definition: $\mathfrak{A} \in TM$ heißt **linear beschränkt**, wenn $p \geq 1$ existiert, so daß $w \in L(\mathfrak{A})\backslash\{\varepsilon\}$ $bv_{\mathfrak{A}}(w) \leq p|w|$

Korollar: $\mathfrak{L}_1(\Sigma) = \mathfrak{L}(\Sigma, lbTM)$
Grund: Simulationen erhalten linearen Platzbedarf/Bandverbrauch.

Definition: $\mathfrak{A} \in LBA(\Sigma) :\Longleftrightarrow \mathfrak{A} lbTM$ mit $p = 1$. Das heißt nur Eingabefelder werden benutzt.

Satz: $\mathfrak{L}_1(\Sigma) = \mathfrak{L}(\Sigma, LBA)$
Beweisidee: Kompression des benutzten Bandbereichs durch Vergrößerung des Arbeitsalphabetes. $\Gamma^L := \Gamma^P$



Deterministische TM, k-Band-TM

Reduktion nicht-deterministischer TM aud deterministische TM in 2 Schritten:
1) Simulation einer nicht-deterministischen TM durch 3-Band dTM.
2) Reduktion von k-Band-dTM auf 1-Band-dTM.

Definition: (deterministische k-Band-TM)
$\mathfrak{A} =< Q, \Sigma, \Gamma, q_0, B, F, \delta >\in$ k-dTM, wenn $\delta : Q \times \Gamma^k - \to Q \times (\Gamma \times \{L, N, R\})^k$ und sonst wie TM.

Konfiguration: $Q \times (\Gamma^* \times \Gamma \times \Gamma^*)^k$
Einzelschrittrelation: simultanes Arbeiten auf k-Bändern gemeinsames δ
Anfangskonfiguration: für $w \in \Sigma^*$

$$K_w := \begin{cases} (q_0, (\varepsilon, a, v), (\varepsilon, \Box, \varepsilon)^{k-1}) & \text{falls } w = av \\ (q_0, (\varepsilon, \Box, \varepsilon)^k) & \text{falls } w = \varepsilon \end{cases}$$

$L(\mathfrak{A}) := \{w \in \Sigma^* \,|\, K_w \vdash^* (q, \dots) \text{ mit } q \in F\}$

Satz: Zu jedem $\mathfrak{A} \in$ TM läßt sich eine äquivalente $\mathfrak{A}' \in$ 3-dTM konstruieren.
Beweis: Für $\delta_{\mathfrak{A}} : Q \times \Gamma \longrightarrow \mathfrak{P}_f(Q \times \Gamma \times \{L, N, R\}$ gelte $\max\{|\delta(q, x)| \,|\, (q, X) \in Q \times \Gamma\} =: r$.
Bezeichnung der Alternativen von $\mathfrak{A}$ durch Elemente von $[r] := \{1, \dots, r\}$
Berechnung von $\mathfrak{A}$ bestimmt $\zeta \in [r]^*$.
Umgekehrt: Auswahl einer Berechnung durch ζ als Steuerwort.
Band 1: Eingabe, wiederholtes Lesen
Band 2: Erzeugung der Zahlenfolgen $\zeta \in [r]^*$
Band 3: Simulation von $\mathfrak{A}$ gemäß Band 2
Sukzessive Berechnung aller $\zeta \in [r]^*$

Für jedes ζ Kopie der Eingabe auf Band 3.
Beachte: Nicht jedes ζ entspricht einer Berechnung.

Satz: Zu jeder $\mathfrak{A} \in$ k-dTM läßt sich ein äquivalentes $\mathfrak{A}' \in$ 1-dTM konstruieren.

Korollar: $\mathfrak{L}(\Sigma, \mathrm{TM}) = \mathfrak{L}(\Sigma, \mathrm{dTM})$

Bemerkung: Für *LBA*'s ist diese Frage noch Offen.[1]

4.3 Aufzählbare und entscheidbare Sprachen

Intuitiv: $L \subseteq \Sigma^*$ ist **aufzählbar**, wenn es ein effektives Verfahren gibt, welches genau die Elemente von L erzeugt.
$L \subseteq \Sigma^*$ ist **entscheidbar**, wenn es effektives Verfahren gibt, welches für jedes $w \in L$ oder $w \notin L$ gibt.

Folgerung: Für $L \subseteq \Sigma^*$ gilt:
L entscheidbar $\Longleftrightarrow$ L und $\Sigma^* \backslash L$ aufzählbar.

Formalisierung: TM mit Ausgabe

Satz: $\{L \subseteq \Sigma^* \mid L \text{ aufzählbar}\} = \mathfrak{L}(\Sigma, \mathrm{dTM})$
Church-Turing-These $\Longrightarrow$ Jede effektiv beschreibbare Sprache ist eine Typ 0-Sprache.

Satz: $\mathfrak{L}_0 \subsetneq \mathfrak{P}(\Sigma^*)$
Beweis: Die Menge der Typ 0-Grammatiken über Σ läßt sich abzählen (nach ihrer Größe). Als ist $\mathfrak{L}_0(\Sigma)$ abzählbar. $\mathfrak{P}(\Sigma^*)$ ist jedoch überabzählbar.

Diagonalverfahren nach Cantor

$|\Sigma| = 1 \quad \Sigma^* = \mathbb{N} \quad L \subseteq \Sigma^* \mathsf{char}_L : \mathbb{N} \longrightarrow \{0,1\}$

Angenommen, $\mathfrak{P}(\mathbb{N})$ ist abzählbar.
$f : \mathbb{N}^2 \longrightarrow \{0,1\}$
$f(i,j) = 1 \Longleftrightarrow \mathsf{char}_{L_i}(j) = 1$
Definiere: $L_{\mathsf{diag}} \subseteq \mathbb{N}$ durch $\mathsf{char}_{L_{\mathsf{diag}}}(j) := 1 - f(j,j)$

	0	1	2	3	...
0	0	1	1	0	
1	1	0	1	0	
2	1	1	0	1	
3					
⋮					

dann kann L_{diag} nicht in der Abzählung vorkommen.

Satz: $DEC(\Sigma) := \{L \subseteq \Sigma^* \mid L \text{ entscheidbar}\} \subseteq \mathfrak{L}_0(\Sigma)$
Beweis: Die Spache Univ. ist nicht entscheidbar. Diagonalisierungsschluß. (Selbstanwendungsproblem, vergleiche Vorlesung "Berechnbarkeit und Komplexität)

[1] Also wer berümt werden will, sollte sich an die Lösung machen.

Satz: Jede Typ-1 Sprache ist entscheidbar.
Beweis: Sei G eine Typ-1 Grammatik über Σ und $w \in \Sigma^*$
Fall 1: $w = \varepsilon \in L \iff S \longrightarrow \varepsilon \in P$
Fall 2: $w \neq varepsilon \in L \iff$ es existiert Ableitung $S \curvearrowright \alpha_1 \curvearrowright \alpha_2 \ldots \curvearrowright \alpha_n = w$ mit $|\alpha_i| \leq |w|$. Es gibt nur endlich viele solche Ableitungen.

Satz: Es gibt entscheidbare, nicht kontextsensitive Sprachen.
Beweis: Literatur

4.4 Unentscheidbare Probleme

Ziel: Die folgenden Probleme sind unentscheidbar:

- das Wortproblem für Typ-0 Beschreibung
- das ϕ-Problem für Typ-1 Beschreibung
- das Äquivalenzproblem für Typ-2 Beschreibung

Bemerkung: Das Äquivalenz-Problem für *DPDA*s wurde kürzlich als entscheidbar nachgewiesen.

Hilfsmittel: Das Postsche Korrespondenzproblem (Vorlesung Berechnbarkeit und Komplexität)
Seien $w = (w_1, w_2, \ldots, w_n)$ und $v = (v_1, v_1, \ldots, v_n)$ mit $w_i, v_i \in \Sigma^*$ und $1 \leq i \leq n$.
Beispiel: Eingabe $(1, 10, 011)$ und $(101, 00, 11)$ Indizes $(1, 3, 2, 3)$
101110011
101110011
Dann heißt $(i_1, \ldots, i_n) \in \{1, \ldots, n\}^k$ **eine Lösung von** $PCP(w, v)$**, falls** $w_{i_1} w_{i_2} \ldots w_{i_k} = v_{i_1} v_{i_2} \ldots v_{i_k}$

Es gilt: Das *PCP* ist unentscheidbar, das heißt es gibt kein Verfahren, welches für $n \in \mathbb{N}$ und $w, v \in (\Sigma^*)^n$ feststellt, ob $PCP(w, v)$ eine Lösung besitzt oder nicht.
Beweis: Reduktion des Halteproblems für TM auf das *PCP*.

Satz: Das ϕ-Problem von Typ-1 Grammatiken ist unentscheidbar.
Beweis: Reduktion des *PCP* auf das ϕ-Problem (Typ 1).
Sei $w = (w_1, \ldots, w_n)$, $v = (v_1, \ldots, v_n) \in (\Sigma^*)^n$ **Konstruktion von** $\mathfrak{A}(w, v) \in$ **lbTM mit** $PCP(w, v)$ **lösbar** $\iff L(\mathfrak{A}(w, v)) \neq \phi$
$\mathfrak{A}(w, v)$ **erzeugt bei Eingabe von** $u \in \Sigma^+$ **alle Folgen** $(i_1, \ldots, i_k) \in \{1, \ldots, n\}^k$ **mit** $k \leq |u|$ **und prüft jeweils, ob** $w_{i_1} \ldots w_{i_k} = v_{i_1} \ldots v_{i_k}$
$\mathfrak{A}$ **erkennt** u**, falls eine solche Indexfolge auftritt. Platzbedarf linear in** $|u|$**.**
Wäre ϕ-Problem entscheidbar, so auch *PCP*.

Index