Inhaltsverzeichnis

Kapitel 1

Alphabete, Wörter, Sprachen

Zeichenreihen als Grundobjekte in der Informatik

1. Grundbegriff

Σ : Alphabet, nicht-leere, endliche Menge

a $\in \Sigma$: Buchstabe, Charakter, Symbol

2. Grundbegriff

Σ^* : Menge der Wörter über Σ

$\Sigma^* := \{a_1, a_2, ..., a_n \mid n \in \mathbb{N}, a_i \in \Sigma\}$
Wörter, Zeichenreihen, Zeichenketten, Strings
n = 0 : das leere Wort, Bez.: ϵ

Operationen auf Σ^*:

- *Verkettung* (Konkatenation)
  $\_ \cdot \_ \ : \ \Sigma^* \times \ \Sigma^* \to \Sigma \qquad (\mathrm{w}\ ,\ \mathrm{v}) \ \mapsto \mathrm{w} \cdot \mathrm{v} := \mathrm{wv}$

  Es gilt:

  1. $\cdot$ ist assoziativ: $(\mathrm{u} \cdot \mathrm{v}) \cdot \mathrm{w} = \mathrm{u} \cdot (\mathrm{v} \cdot \mathrm{w})$
  2. ϵ ist $\cdot$ - (verkettungs-) neutral : $\epsilon \cdot \mathrm{w} = \mathrm{w} \cdot \epsilon = \mathrm{w}$
     Sprechweise: $< \Sigma^* \ ; \ \_ \cdot \_ \ ; \ \epsilon >$ ist eine Halbgruppe mit Eins (Monoid) .
     Bemerkung: freie Erzeugung.

- *Länge eines Wortes:* $\mathrm{w} = a_1 a_2 ... a_n \in \Sigma^*, \ |a_1 ... a_n| := \mathrm{n}$, falls alle $a_i \in \Sigma$,
  also: $|\epsilon| = 0$ *und* $|wv| = |w| + |v|$

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

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

3. Grundbegriff:

$p(\Sigma^*)$: Menge der sogenannten formalen Sprachen über Σ. Eine (formale) Sprache (über Σ) ist eine beliebige Teilmenge von Σ^*. Es bestehen folgende Beschreibungsmöglichkeiten für Sprachen: Automaten und Grammatiken.
(Potenzmenge ≡ Menge aller Teilmengen)
$p(\Sigma^*) := \{L \mid L \subseteq \Sigma^*\}$

Beispiel: $\emptyset, \{\epsilon\}, \{w_1, w_2, ..., w_n\}, \Sigma^*$

Operationen auf $p(\Sigma^*)$

- Boolesche Operation: $L_1 \cup L_2,\ L_1 \cap L_2,\ \overline{L} := \Sigma^* \setminus L$

- Komplexprodukt (jeder mit jedem): $L_1L_2 := \{w_1w_2 \mid w_i \in L_i\}$
  $LL := \{w_1w_2 \mid w_i \in L\}$

- Potenzen einer Sprache:
  $L^0 := \{\epsilon\}$ (weil $\{\epsilon\}L = L$)
  $L^{n+1} := LL^n$

- Stern einer Sprache (Iteration / Repitition) (unendliche Sprache)

$$L^* := \bigcup_{n\in\mathbb{N}} L^n \quad \text{(incl. } \epsilon\text{)}$$

$$L^+ := LL^* = \bigcup_{n=1}^{\infty} \quad \text{(excl.}\epsilon\text{)}$$

- Spiegelbild einer Sprache
  $L^R := \{w^R \mid w \in L\}$

- reguläre Operationen
  $L_1 \cup L_2,\ L_1L_2,\ L^*$

Kapitel 2

Reguläre Sprachen

2.1 Reguläre Ausdrücke

endliche Beschreibungen unendlicher Sprachen
wesentliches Hilfsmittel: 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 $\alpha \in RA(\Sigma)$

Vereinfachte Schreibweisen

- 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:
$[\![\_]\!] : RA(\Sigma) \to p(\Sigma^*)$

- $[\![\Lambda]\!] := \emptyset$
- $[\![a]\!] := \{a\}$
- $[\![(\alpha \vee \beta)]\!] := [\![\alpha]\!] \vee [\![\beta]\!]$
- $[\![\alpha \cdot \beta]\!] := [\![\alpha]\!][\![\beta]\!]$ (Komplexprodukt)
- $[\![\alpha^*]\!] := [\![\alpha]\!]^*$

Die so beschreibbaren Sprachen heißen *regulär*.

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

- $\emptyset, \{a\} \in REG(\Sigma) \; \forall \; a \in \Sigma$
- $L, L' \in REG(\Sigma) \curvearrowright L \cup L', \; LL', \; L^* \in REG(\Sigma)$

Anwendungen:

- Dateiauswahl: ls *.ps
- Textsuche: egrep -E 'Gr(ae—ä)del' publications.txt
- Suchmaschine: and, not
- Symbolklassen Fließkomma in C
  - ExpPart [eE][-+]?[0-9]+
  - FractConst ([0-9]*"·"[0-9]+)—([0-9]+"·)
- WHILE-Programme : formale Pfadsprachen

Die Menge WP der WHILE_0-Programme (Prof. Thomas) ist induktiv aufgebaut über den Mengen
$A := \{X_i := X_i + 1, X_i := X_i - 1 \mid i \in \mathbb{N}\}$ und
$B := \{X_i > 0 \mid i \in \mathbb{N}\}$ durch:

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

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

Für $P \in WP$ definieren wir die *formale Pfadsprache* $L_p \subseteq (\Sigma^*_{PATH})$

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

es folgt: Beschreibung von L_P durch $\alpha_P \in RA(\Sigma_{PATH})$, nämlich:

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

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

Definition $P \underset{PATH}{\sim} Q :\curvearrowleft \ L_P = L_Q$

Folgerung: $P \underset{PATH}{\sim} Q :\curvearrowright P \sim Q$ (d.h. $f_P = f_Q$)
$\downarrow$ $\downarrow$
entscheidbar durch endl. Automaten — unentscheidbar

$\alpha \mapsto [\![\alpha]\!] \subseteq \Sigma^*$
$L(\alpha) := [\![\alpha]\!]$
$\mathcal{L}(\Sigma, RA) = REG(\Sigma)$

Standardprobleme für $RA(\Sigma)$:

- Wortproblem (matching Problem)
  Gilt $w \in [\![\alpha]\!]$ für $w \in \Sigma^*$ und $\alpha \in RA(\Sigma)$
- Äquivalenzproblem
  Gilt $[\![\alpha]\!] = [\![\beta]\!]$ für $\alpha, \beta \in RA(\Sigma)$
- Leerheitsproblem
  Gilt $[\![\alpha]\!] = \emptyset$ für $\alpha \in RA(\Sigma)$

Alle Probleme sind mit endlichen Automaten entscheidbar.

2.2 Deterministische endliche Automaten (DFA)

Definition: Seien Q und Σ nicht-leere, endliche Mengen.
$q_0 \in Q, F \subseteq Q$ und $\delta : Q \times \Sigma \to Q$
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 Endzustandsmenge F
Darstellung von $\mathfrak{A}$ durch Transitionstafel und Zustandgraphen.

Definition: $\mathfrak{A} =< Q, \Sigma, \delta, q_0, F >\in DFA(\Sigma)$ bestimmt die erweiterte Transitionsfunktion
$\overline{\delta} : Q \times \Sigma^* \to Q$ mit
$\overline{\delta}(q, \epsilon) := q$
$\overline{\delta}(q, wa) := \delta(\overline{\delta}(q, w), a)$
(* *oder:* $\overline{\delta}(q, aw) := \overline{\delta}(\delta(q, a), w)$ *)
und damit die von $\mathfrak{A}$ erkannte **Sprache**: $L(\mathfrak{A}) := \{w \in \Sigma^* \mid \overline{\delta}(q_0, w) \in F\}$

Ziel: $\mathcal{L}(\Sigma, DFA) = REG(\Sigma)$
Hilfsmittel: nicht deterministische Automaten!

2.3 Nicht deterministische endliche Automaten (NFA)

Def. Seien Q, Σ, q_0, F wie bei einem $\mathfrak{A} \in DFA$
ferner $\Sigma_\epsilon := \Sigma \cup \epsilon$ und $\delta : Q \cup \Sigma_\epsilon \to p(Q)$

Dann heißt $\mathfrak{A} =< Q, \Sigma, \delta, q_0, F >$ ein *nicht deterministischer endl. Automat über* Σ

Es folgt: $DFA(\Sigma) \subseteq NFA(\Sigma)$
(Hinweis: Worttransitionen sind durch Zwischenzustände simulierbar)

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

1. Σ-Transitionen: $(q, aw) \vdash (q', w) \quad \curvearrowright \quad q' \in \delta(q, a)$
2. ϵ-Transitionen: $(q, w) \vdash (q', w) \quad \curvearrowright \quad q' \in \delta(q, \epsilon)$

Dann ist die durch $\mathfrak{A}$ erkannte **Sprache** definiert als:
$L(\mathfrak{A}) := \{w \in \Sigma^* | (q_0, w) \vdash^* (q, \epsilon)$ mit $q \in F\}$

Ziel:
$\alpha \in RA(\Sigma) \;\mapsto\; \mathfrak{A}(\alpha) \in NFA(\Sigma) \;\mapsto\; \mathfrak{A}(\alpha)_P \in DFA(\Sigma)$ mit
$[\![\alpha]\!] \;=\; L(\mathfrak{A}(\alpha)) \;=\; L(\mathfrak{A}(\alpha)_P)$

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

- $Q_P := \{T \subseteq Q \mid \exists w \in \Sigma^* : T = \{q \in Q \mid (q_0, w) \vdash^* (q, \epsilon)\}$ (Zustände von $\mathfrak{A}_P)\}$
- $q_{0_P} := \{q \in Q \mid (q_0, \epsilon) \vdash^* (q, \epsilon\}$
- $F_p := \{T \in Q_P \mid T \cap F \neq \emptyset\}$
- $\delta_P : Q_P \times \Sigma \to Q_P$
- $\delta_P(T, a) := \{q' \in Q \mid (q, a) \vdash^* (q', \epsilon)$ für ein $q \in T\}$

Beachte:
$T \subseteq Q_P$
$\curvearrowright \; \exists w \in \Sigma^* : T = \{q' \in Q \mid (q_0, w) \vdash^* (q', \epsilon)\}$
$\curvearrowright \; \delta_P(T, a) = \{q' \in Q \mid (q_0, wa) \vdash^* (q', \epsilon)\} \in Q_P$

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:
$\underbrace{q \in \overline{\delta}_P(q_{0_P}, w)}_{DFA} \;\curvearrowright\; \underbrace{(q_0, w) \vdash^* (q, \epsilon)}_{NFA}$ (q liegt im Folgezustand des neuen PM-Automaten)

Induktion:
(i) $\underline{w = \epsilon}$:
$q \in \overline{\delta}(q_{0_P}, \epsilon)$
$\curvearrowright \; q \in q_{0_P}$
$\curvearrowright \; (q_0, \epsilon) \vdash^* (q, \epsilon)$
(ii) $\underline{w = w'a}$:
$q \in \overline{\delta}_P, w'a) = \delta_P(q_{0_P}, w'), a)$
$\curvearrowright \; \exists \; q' \in \overline{\delta}_P(q_{0_P}, w') : (q', a) \vdash (q, \epsilon)$
$\curvearrowright \; \exists \; q' \in Q : (q_0, w'a) \vdash^* (q', a) \vdash^* (q, \epsilon)$
$\curvearrowright \; (q_0, w'a) \vdash^* (q, \epsilon)$

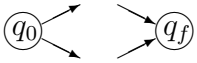
Daraus folgt die Behaupung:

$w \in L(\mathfrak{A}) \;\curvearrowright\; \exists \; q \in F : (q_0, w) \vdash^* (q, \epsilon)$
$\curvearrowright \; q \in \overline{\delta}_P(q_{0P}, w)$ *und* $q \in F$
$\curvearrowright \; \overline{\delta}_P(q_{0P}, w) \in F \qquad \curvearrowright \; w \in L(\mathfrak{A}_P)$ $\diamondsuit$

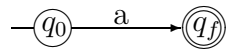
2.4 Synthese und Analyse endlicher Automaten

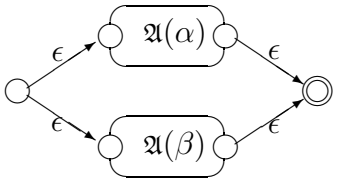
Synthese:
Konstruiere für $\delta \in RA(\Sigma)$ einen äquivalenten $\mathfrak{A}(\delta) \in NFA$, d.h. $[\![\delta]\!] = L(\mathfrak{A}(\delta))$

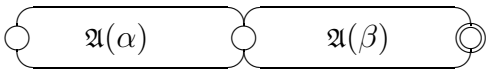
Der Algorithmus von Thompson
Idee: $\mathfrak{A}(\delta)$ hat genau einen Endzustand q_f,
q_0 ist Quelle, q_f ist Senke (im Zustandsgraphen)

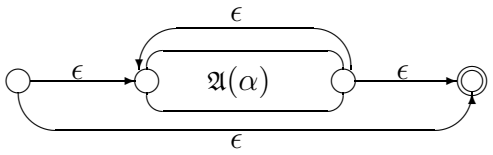


- $\mathfrak{A}(\Lambda)$:= 

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

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

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

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

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

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

Analyse:
Konstruiere für $\mathfrak{A} \in DFA(\Sigma)$ einen $\alpha(\mathfrak{A}) \in RA(\Sigma)$ mit $[\![\alpha(\mathfrak{A})]\!] = L(\mathfrak{A})$
$\mathfrak{A} =< Q, \Sigma, \delta, q_1, F > \in DFA(\Sigma)$ und $Q = \{q_1, ..., q_n\}$
Für $i, j \in \{1, ..., n\}$ und $k \in \{0, 1, ..., n\}$ definieren wir
$W_{ij}^k := \{w \in \Sigma^* |$ w überführt q_i in q_j ohne Benutzung von $q_{k+1}, ..., q_n$ als Zwischenzustände$\}$.

Dann gilt:

$$L(\mathfrak{A}) = \bigcup_{q_j \in F} W_{1j}^n$$

Somit genügt der Nachweis der Regularität der Sprachen W_{ij}^k

Induktion über k:
i) k=0: $W_{ij}^0 \subseteq \Sigma_\epsilon$ ist regulär ($\{\epsilon\} = [\![\Lambda^*]\!]$)
ii) k-1→k : $W_{ij}^k = W_{ij}^{k-1} \cup W_{ik}^{k-1}(W_{kk}^{k-1})^* W_{kj}^{k-1}$

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

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

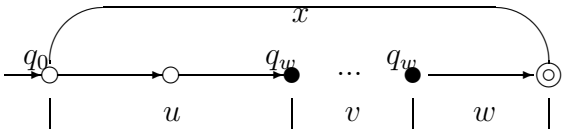
Beweisidee: a) Komplement: $F' := Q \setminus F$
b) $L_1 \cap L_2 \;=\; \overline{L}_1 \cup \overline{L}_2$

2.5 Das Pumping - Lemma

Hilfsmittel zum Nachweis nicht-regulärer Sprachen
Satz: (Pumping-Lemma, Iterationslemma)
Sei $L \in REG(\Sigma)$, Dann $\exists\, k \in \mathbb{N}$, 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. $|uv^i w| \in L$ für alle $i \in \mathbb{N}$, insbesondere für $i = 0$

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



Im 1. Abschnitt wurde u erkannt
Im 2. Abschnitt wurde v erkannt
Im 3. Abschnitt wurde w erkannt

Seien $\overset{q_w}{\bullet}$ $\overset{q_w}{\bullet}$ das erste Wiederholungspaar von Zuständen, so folgen die Eigenschaften 1 - 3 ♢

Beachte:
Das Pumping-Lemma beschreibt eine notwendige, aber nicht eine hinreichende Eigenschaft regulärer Sprachen, daher ist es geeignet zum Nachweis nicht-regulärer Sprachen.

Beispiel: $L = \{a^n b^n | n \geq 1\} \notin REG(\{a, b\})$

Beweis: Angenommen, $L \in REG(\Sigma)$. Dann $\exists\ k \in \mathbb{N}$ mit den Eigenschaften des Pumping-Lemmas, "Pumping-Index"
Für $x = a^k b^k$ muß dann eine Zerlegung existieren $a^k b^k = uvw$ mit $v \neq \emptyset$ und $|uv| \leq k$, also $v \in \{a^i \mid i > 0\}$ und $uw = a^{k-|v|} b^k \in L$ Widerspruch! $\diamondsuit$

2.6 Zustandsreduktion endlicher Automaten

Ziel: Konstruktion endlicher Automaten mit minimaler Zustandsanzahl

Definition: Für $L \subseteq \Sigma^*$ und $w \in \Sigma^*$ heißt $d_w(L)$ ($d_w(L)$ ist das, was auf w in L folgt)
$d_w(L) := \{v \in \Sigma^* \mid wv \in L\}$ 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 $D(L) = \{d_w(L) \mid w \in \Sigma^*\}$ $|D(L)| \leq |Q|$ (D(L) ist Menge aller Ableitungen)

Beweis:
Für $q \in Q$ bezeichne $L(q) := L(< Q, \Sigma, \delta, q, F >)$. Dann gilt:

$$d_w(L) \ = \ d_w(L(q_0)) \ = \ L(\overline{\delta}(q_0, w)) \qquad \diamondsuit$$

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

- $q_0 := d_\epsilon(L) = L$
- $F := \{d_w(L) \mid w \in L\}$ (d.h. $\epsilon \in d_w(L)$)
- $\delta(d_w(L), a) := d_{wa}(L)$ (Instanz von $\delta(q, a) = q'$)

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

Lemma: $L(\mathfrak{A}_L) = L$

Beweis:

$$w \in L(\mathfrak{A}_L) : \curvearrowright \ \overline{\delta}(q_0, w) \in F \ \curvearrowright \ d_w(L) \in F \ \curvearrowright \ w \in L \qquad \diamondsuit$$

Korollar:

1. Der Ableitungsautomat ist zustandsminimal
2. Für $L \subseteq \Sigma^*$ gilt: $L \in REG \quad \curvearrowleft \quad |D(L)| < \infty$

Zustandsreduktion
Konstruktion einses zustandsminimalen Automaten aus einem gegebenen Automaten durch:

- Weglassen nicht erreichbarer Zustände
- Verschmelzen äquivalenter Zustände

Dann heißt:

- $q \in Q$ *erreichbar* : $\curvearrowleft \ \exists w \in \Sigma^* : \overline{\delta}(q_0, w) = q$
- $q_1 \sim q_2$ (äquivalent) : $\curvearrowleft \ L(q_1) = L(q_2)$
  Wenn die Menge der Beschriftungen gleichist , dann ist auch die Menge der Zustände gleich

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

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

$\overline{\delta}(q, \epsilon) := q$

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

Lemma: Für $\underline{\mathfrak{A}} \in DFA(\Sigma)$ gilt (bis auf Zustandsnamen) :

$$\mathfrak{A}/\sim \ \cong \ \mathfrak{A}_{L(\underline{\mathfrak{A}})}$$

Insbesondere ist $\mathfrak{A}/\sim$ äquivalent zu $\mathfrak{A}$, und es folgt:
Zustandsminimale Automaten sind bis auf Isomorphie eindeutig bestimmt.

Beweis: Sei $\beta : \tilde{Q} \to D(L(\mathfrak{A}))$ definiert durch $\beta([\overline{\delta}(q_0, w)]) := d_w(L(\mathfrak{A}))$
β ist unabhängig vom Repräsentanten $w \in \Sigma^*$:

$$\overline{\delta}(q_0, v) \sim \overline{\delta}(q_0, w) \ \curvearrowright \ L(\overline{\delta}(q_0, v)) = L(\overline{\delta}(q_0, w) \ \curvearrowright \ d_w(L(\mathfrak{A})) = d_v(L(\mathfrak{A}))$$

- β ist bijektiv, weil die obige Folgerung umkehrbar ist.
- β ist strukturerhaltend:
  - $\beta([q_0]) = \beta([\overline{\delta}(q_0, \epsilon)]) = d_\epsilon(L(\mathfrak{A}))$
    Anfangszustand des Ableitungsautomaten
  - $\overline{\delta}(q_0, w) = q \in F : \beta([\overline{\delta}(q_0, w)]) = d_w(L(\mathfrak{A}))$
    Endzustand des Ableitungsautomaten
  -
$$\begin{array}{rrcl} & \tilde{\delta}([q], a) & = & [\delta(q, a)] \\ \curvearrowright & \tilde{\delta}([\overline{\delta}(q_0, w)], a) & = & [\overline{\delta}(q_0, wa)] \\ \curvearrowright & \delta(d_w(L(\mathfrak{A})), a) & = & d_{wa}(L(\mathfrak{A})) \end{array}$$

$\diamondsuit$

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 ($q_1 \overset{k}{\sim} q_2$): $\curvearrowright \ \forall w \in \Sigma^*, |w| \leq k : w \in L(q_1) \ \curvearrowright \ w \in L(q_2)$
- $\mathfrak{A}$ induziert die Abbildungen $R^k : Q^k \to \{0, 1\}$ mit $r^k(q_1, q_2) = 1 : \curvearrowright \ q_1 \overset{k}{\sim} q_2$
  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-Äquivalenzen

- $q_1 \overset{0}{\sim} q_2 \ \curvearrowright \ (q_1 \in F \ \curvearrowright \ q_2 \in F)$
- $q_1 \overset{k+1}{\sim} q_2 \ \curvearrowright \ q_0 \overset{0}{\sim} q_2$ und $\delta(q_1, a) \overset{k}{\sim} \delta(q_2, a) \forall a \in \Sigma$

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

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

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 $q \neq q'$
- Markiere alle Paare $\{q, q'\}$ mit $q \in F$ und $q' \notin F$ oder umgekehrt
- Für jedes unmarkierte Paar $\{q, q'\}$ und Eingabesymbol $a \in \Sigma$, teste ob $\{\delta(q, a), \delta(q', a)\}$ markiert. Wenn ja, markiere $\{q, q'\}$.
- Wiederhole letzten Schritt, bis keine Änderung mehr erfolgt.
- Unmarkierte Paare repräsentieren äquivalente Zustände.

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

2.7 Entscheidbare Eigenschaften

2.7.1 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.

- $\emptyset$-Problem: $L(\mathfrak{A}) = \emptyset$? für $\mathfrak{A} \in DFA(\Sigma)$
  Durch Eingabe aller Werte w mit $|w| < |Q|$ entscheidbar.
  Beachte: $w \in L(\mathfrak{A}), |w| \geq |Q| \curvearrowright \exists v \in L(\mathfrak{A}), |v| < |w|$
  Verfahren: testen, ob F von q_0 erreichbar ist.
  Durchführung in $O(|Q|^2)$-Zeit. Bei geschickter Implementierung der Mengenoperationen auch besser.

- $\sim$-Problem: $L(\mathfrak{A}) = L,\ (\mathfrak{A}')$? für $\mathfrak{A}\mathfrak{A}' \in DFA(\Sigma)$
  Reduktion auf das $\emptyset$-Problem:
  $L(\mathfrak{A}) = L(\mathfrak{A}') \curvearrowright L(\mathfrak{A}) \subseteq L(\mathfrak{A}')$ und $L(\mathfrak{A}') \subseteq L(\mathfrak{A})$
  $\curvearrowright \underbrace{L(\mathfrak{A}) \cap \overline{L(\mathfrak{A}')}}_{\mathfrak{A}_1}$ und $\underbrace{L(\mathfrak{A}') \cap \overline{L(\mathfrak{A})}}_{\mathfrak{A}_2} = \emptyset$
  (Komplementautomat $\equiv$ Komplement der Endzustände)
  2 Produktautomaten mit $|Q||Q'|$ Zuständen auf $\emptyset$ testen.
  $\sim$-Problem ist in $O(|Q|^2|Q|^2)$ lösbar.

2.7.2 Reguläre Ausdrücke, nicht-deterministische Automaten (NFA)

Durch Transformationen in DFA's sind alle drei Probleme entscheidbar.
Aber der Aufwand steigt:

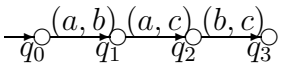
- Wortproblem: $O(|\alpha||w|)$-Zeit bzw. $O(|\alpha||w|)$-Platz
- $\emptyset$-Problem: NFA wie DFA behandeln. $RA \mapsto NFA$ in $O(|\alpha|)$-Zeit mit $|Q| \leq 2(\alpha)$
- $\sim$-Problem: NP-hart (Problem $\in NP \Rightarrow P \leq_p$, $\sim$-Problem)

2.8 endliche Automaten mit Ausgabe

Idee: Bei Zustandsübergängen erfolgt sequentielle Ausgabe

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

Bsp: $f(aab) = bcc$



2.9 Automaten als abstrakte Programme

2.9.1 GOTO-Programme

GOTO-Programme sind aufgebaut über den Mengen
$A := \{X_i := X_i + 1, X_i := X_i - 1 \mid i \in \mathbb{N}\}$ (Anweisungen)
und $B := \{X_i = 0 \mid i \in \mathbb{N}\}$ (Bedingungen)
nämlich als Zeichenreihen der Form $1 : \alpha_1;\ 2 : \alpha_2;\ ...;\ k : \alpha_k$ mit
$\alpha_i \in A \cup \{\underline{if}\ b\ \underline{then}\ j_0\ \underline{else}\ j_1 \mid b \in B, 1 \leq j_0, j_1 \leq k\}$ und $\alpha_k = \text{STOP}$
Sei $\Sigma_{PATH} := A \cup (B \times \{true, false\})$. Für ein GOTO-Programm $P = 1 : \alpha_1,\ ...,\ k : STOP$
definieren wir ein $\mathfrak{A}_P =< Q, \Sigma_{PATH}, \delta, q_0, F > \in DFA(\Sigma)$

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

Definition: $L_P := L(\mathfrak{A}_P)$ heißt formale Pfadsprache von P

$P \underset{PATH}{\sim} Q :\curvearrowleft\ L_P = L_Q$

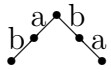
Lemma: $P \underset{PATH}{\sim} Q \ \curvearrowright\ P \sim Q$ (aber nicht umgekehrt)

entscheidbar unentscheidbar

2.9.2 Parallele Programme, Verteilte Systeme

Reduktion von Parallelität auf Nichtdeterminismus.

Für Aktionen a und b wird $a \parallel b := ab \vee ba$



NFA zur Modellierung verteilter Systeme

Beispiel: Deadlocks als Senkzustände *Bsp. MS-Windows*

Kapitel 3

Kontextfreie Grammatiken und Kellerautomaten

3.1 Kontextfreie Grammatiken

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

Bezeichnungen und Konventionen

$A, B, C, ...$	$\in$	N	Nichtterminalsymbole
$a, b, c, ...$	$\in$	Σ	Terminalsymbole
S	$\in$	N	Startsymbol
$X, Y, Z, ...$	$\in$	$\mathcal{X}$	$:= N \cup \Sigma$ (Symbole der Grammatik)
$\alpha, \beta, \gamma, ...$	$\in$	$\mathcal{X}^*$	Satzformen
$u, v, w, ...$	$\in$	Σ^*	Terminalwörter
$A \to \alpha := (A, \alpha)$	$\in$	P	Regel, Produktion

Definition: Sei $\mathcal{G} =< N, \Sigma, P, S > \in$ CFG und $\pi = A \to \alpha \in P$.
π bestimmt eine *Ableitungsrelation*: $\Rightarrow_\pi \subseteq \mathcal{X}^* \times \mathcal{X}^*$
mit $\beta_1 \Rightarrow_\pi \beta_2 \curvearrowright$ es existiert ein Kontext $\gamma_1, \gamma_2 \in \mathcal{X}^*$,
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*
$\mathcal{G}$ bestimmt die *Ableitungsrelation* $\Rightarrow_\mathcal{G} \subseteq \mathcal{X}^* \times \mathcal{X}^*$ durch

$$\Rightarrow_\mathcal{G} := \bigcup_{\pi \in P} \Rightarrow_\pi$$

und damit die *von $\mathcal{G}$ erzeugte Sprache* $L(\mathcal{G}) := \{w \in \Sigma^* \mid S \overset{*}{\Rightarrow}_{\mathcal{G}} w\}$

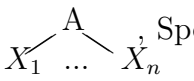
$$\overset{*}{\Rightarrow}_{\mathcal{G}} := \bigcup_{n=0}^{\infty} \overset{n}{\Rightarrow}_{\mathcal{G}} \qquad \text{(reflexive und transitive Hülle)}$$

Es folgt: $w \in L(\mathcal{G}) \quad \curvearrowright \quad \exists \alpha_0, \alpha_1, ..., \alpha_n \in \mathcal{X}^*$
und $\pi_1, ..., \pi_n \in P : S = \alpha_0 \Rightarrow_{\pi_1} \alpha_1 ... \Rightarrow_{\pi_n} \alpha_n = w$

Bezeichnung: $\mathrm{CFL}(\Sigma) := \{L \subseteq \Sigma^* \mid \exists \mathcal{G} \in \mathrm{CFG}(\Sigma) : L(\mathcal{G}) = L\}$

Ableitungsbäume, Rechts- und Linksableitungen, Ein- und Mehrdeutigkeit

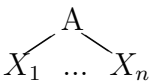
Sei $\mathcal{G} =< N, \Sigma, P, S > \in$ CFG
Darstellung von Ableitungen durch Bäume

1. Eine Regel $\pi = A \to x_1, ...x_n$ bestimmt den *Regelbaum* 

, Spezialfall

für $n = 0$

2. Eine Ableitung $A \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow ... \Rightarrow \alpha_n$ bestimmt den *Ableitungsbaum* durch entsprechendes Verkleben der Regelbäume.
   Definiert durch Induktion über $n \in \mathbb{N}$.

   - $n = 1$ Regelbaum
   - $n \to n+1$ $A \Rightarrow \alpha_1 \Rightarrow ... \Rightarrow \alpha_n$ habe den Ableitungsbaum: 

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

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

Definition: Eine Ableitung heißt Rechtsbleitung (Linksableitung), wenn jeder Ableitungsschritt (α, π, β) ein *Rechtsableitungsschritt*, d.h. $\beta \in \Sigma^*$ (Linksableitungsschritt, d.h. $\alpha \in \Sigma^*$) ist.
Schreibweise: $\underset{r}{\Rightarrow}$ bzw. $\underset{l}{\Rightarrow}$

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

Definition:

- $\mathcal{G} \in \mathrm{CFG}(\Sigma)$ heißt *eindeutig* $:\curvearrowright$ zu jedem w $\in L(\mathcal{G})$ gibt es genau eine Rechtsableitung $S \underset{r}{\Rightarrow} \alpha_1 \underset{r}{\Rightarrow} ... \underset{r}{\Rightarrow} w$

- $\mathcal{G}$ heißt *mehrdeutig* $:\curvearrowright$ nicht eindeutig.
  Beispiel für Mehrdeutigkeit: $a + b * c$

3.2 Einseitig-lineare Grammatiken

Def. Sei $\mathcal{G} =< N, \Sigma, P, S >\in$ CFG.

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

Ziel: $\{L(\mathcal{G}) \mid \mathcal{G}\ linkslinear\}$
$= \{L(\mathcal{G}) \mid \mathcal{G}\ rechtslinear\}$
$= \{L(\mathcal{G}) \mid \mathcal{G}\ einseitig - linear\} \ = \ REG(\Sigma)$

Satz: $\mathcal{L}(\Sigma,\text{NFA}) = \{L \subseteq \Sigma^* | L = L(\mathcal{G}), \mathcal{G}\ rechtslinear\}$

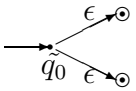
Beweis: 1.) " $\supseteq$ "
Sei $\mathcal{G} =< N, \Sigma, P, S >$ rechtslinear. Auffassung von $\mathcal{G}$ als $\mathfrak{A} =< Q, \Sigma, \delta, q_0, F >\in NFA$
$Q := N \cup \{q_f\}$ $\quad q_f$ neu
$q_0 := S$
$F := \{q_f\}$
$\overline{\delta}(A, w) \ni B$, falls $A \to wB$ in $\mathcal{G}$ existiert.
$\overline{\delta}(A, w) \ni q_f$, falls $A \to wB$ in $\mathcal{G}$ existiert.
Offensichtlich: $L(\mathcal{G}) = L(\mathfrak{A})$
Wortübergänge durch Zwischenzustände beseitigen.
2.) " $\subseteq$ "
$\mathfrak{A} =< Q, \Sigma, \delta, q_0, F >\in NFA$ kann ebenso als rechtslineare Grammatik aufgefaßt werden.
$\diamondsuit$

Korollar: $REG(\Sigma) \subseteq CFL$
Für $|\Sigma| \geq 2$ ist diese Inklusion echt, weil $\{a^n b^n \mid n \in \mathbb{N}\} = L(\mathcal{G})$ mit $\mathcal{G} = (\delta \to aSb \mid \epsilon)$

Bem.: $|\Sigma| = 1 \ \curvearrowright \ L^R \in REG(\Sigma)$

Lemma: $L \in REG(\Sigma) \curvearrowright L^R \in REG(\Sigma)$

Beweis: Zu $\mathfrak{A} \in DFA(\Sigma)$ konstruieren wir $\mathfrak{A}^R \in NFA(\Sigma)$ mit $L(\mathfrak{A}^R) = L(\mathfrak{A})^R$
Idee: ersetze $\overset{a}{\underset{q}{\bullet}}\!\!\rightarrow\!\!\underset{q}{\bullet}$, durch $\underset{q}{\bullet}\!\!\leftarrow\!\!\overset{a}{\underset{q}{\bullet}}$,
$\longrightarrow q_0$ durch $\underset{q_0}{\odot}$

und ergänze diesen NFA durch  für alle $q \in F$ ♢

Korollar: $\{L \subseteq \Sigma^* | L = L(\mathcal{G}), \mathfrak{A} \text{ rechtslinear}\}$
$= \{L \subseteq \Sigma^* | L = L(\mathcal{G}), \mathfrak{A} \text{ linkslinear}\}$

Beweis: L **ist rechtslinear erzeugbar.**
$\curvearrowright L^R$ **ist rechtslinear erzeugbar.**
$\curvearrowright L = L^{RR}$ **ist linkslinear erzeugbar.** ♢

3.3 Normalformen von kontextfreien Grammatiken

Hilfsmittel: Vorgänger von Satzformen

$\alpha \Rightarrow \beta$ α heißt *direkter Vorgänger* von β

$\alpha \overset{*}{\Rightarrow} \beta$ α *Vorgänger* von β

Definition: Sei $\mathcal{G} =< N, \Sigma, P, S > \in CFG$ und $L \subset \mathcal{X}^*$.
Dann ist

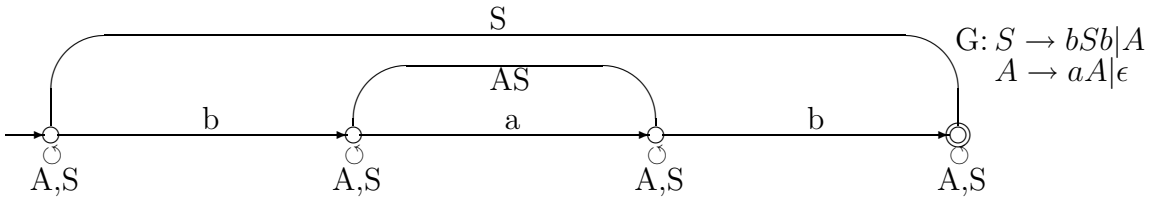
1. die *direkte Vorgängermenge von L* definiert durch:
   $\underline{Pre}_{\mathcal{G}}(L) := \{\alpha \in \mathcal{X}^* \mid \exists \beta \in L : \alpha \Rightarrow_{\mathcal{G}} \beta\}$

2. der *Vorgängerabschluß von L* definiert durch:
   $\underline{Pre}_{\mathcal{G}}(L) := \{\alpha \in \mathcal{X}^* \mid \exists \beta \in L : \alpha \overset{*}{\Rightarrow}_{\mathcal{G}} \beta\}$

Lemma: Sei $\mathcal{G} =< N, \Sigma, P, S > \in CFG$ und $L \subseteq \mathcal{X}^*$.
Dann gilt:
$L \in REG(\mathcal{X}) \ \curvearrowright \ \underline{Pre}_{\mathcal{G}}(L) \in REG(\mathcal{X})$

Beweis: Sei $\mathfrak{A} =< Q, \mathcal{X}, \delta, q_0, F > \in NFA$ mit $L(\mathfrak{A}) = L$
Konstruiere $\mathfrak{A}' =< Q, \mathcal{X}, \delta', q_0, F > \in NFA$ durch hinzufügen folgender Transitionen:
wenn $A \to \alpha \in P$ und $\overline{\delta}(q, \alpha) \ni q'$, so $\overline{\delta}(q, A) \ni q'$.
Ergänze δ um solche Transitionen, solange das möglich ist. Dieser Erweiterungsprozeß terminiert, weil Q und $\mathcal{X}$ endlich sind. Die Korrektheit ist offensichtlich. ♢

Beispiel:



Definition: Sei $\mathcal{G} \in CFG(\Sigma)$ und $A \in N$

1. A heißt *produktiv* $:\curvearrowright \exists w \in \Sigma^* : A \overset{*}{\Rightarrow} w$

2. A heißt *erreichbar* $:\curvearrowright \exists \alpha, \beta \in \mathcal{X}^* : S \overset{*}{\Rightarrow} \alpha A \beta$

Folgerung: A produktiv $\curvearrowright A \in \underline{Pre}_{\mathcal{G}}(\Sigma^*)$
A erreichbar $\curvearrowright S \in \underline{Pre}_{\mathcal{G}}(\mathcal{X}^*\{A\}\mathcal{X}^*)$

Definition: $\mathcal{G} \in CFG(\Sigma)$ heißt *reduziert* $:\curvearrowright$
entweder $P = \emptyset$
oder jedes $A \in N$ ist produktiv und erreichbar.

Lemma: Jedes $\mathcal{G} \in CFG(\Sigma)$ läßt sich in ein äquivalentes reduziertes $\mathcal{G}' \in CFG(\Sigma)$ transformieren.

Beweis: Stelle für jedes $A \in N$ mit Hilfe von NFA fest, ob A erreichbar und produktiv ist.
Fall 1: S nicht produktiv: Wähle $P = \emptyset$
Fall 2: S produktiv: Weglassen aller $A \in N$, die nicht produktiv oder nicht erreichbar sind, sowie aller Regeln, in denen solche A's vorkommen. $\diamondsuit$

Korollar: Das Leerheitsproblem von CFG's ist entscheidbar.

Elimination von ϵ-Regeln

Definition: $\mathcal{G} \in CFG(\Sigma)$ heißt ϵ-frei $:\curvearrowright$
$(A \to \epsilon \in P \curvearrowright A = S$ und S auf keiner rechten Regelseite)

Lemma: $X \overset{*}{\Rightarrow} \epsilon \quad \curvearrowright X \in \underline{Pre}_{\mathcal{G}}(\{\epsilon\})$

Satz: Jedes $\mathcal{G} \in CFG(\Sigma)$ läßt sich in eine äquivalente ϵ-freie Grammatik $\mathcal{G}' \in CFG(\Sigma)$ transformieren.

Beweis: Konstruiere mit $\underline{Pre}_{\mathcal{G}}(\{\epsilon\})$ die Menge $N_\epsilon := \{A \in N | A \overset{*}{\Rightarrow} \epsilon\}$ und damit
$\mathcal{G}' =< N', \Sigma, P', S' > \in CFG$:

- $N' := N \dot{\cup} \{S'\}$

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

Beachte: Wegen $k \geq 1$ entfallen ϵ-Regeln und $\mathcal{G}'$ ist ϵ-frei.

Bleibt zu zeigen: $L(\mathcal{G}') = L(\mathcal{G})$

1. $w = \epsilon$:
   $\epsilon \in L(\mathcal{G}') \curvearrowright S \in N_\epsilon \curvearrowright S \Rightarrow_{\mathcal{G}} \epsilon \curvearrowright \epsilon \in L(\mathcal{G})$

2. $w \neq \epsilon$:

   (a) $w \in L(\mathcal{G}'), d.h. \curvearrowright S \overset{*}{\Rightarrow}_{\mathcal{G}'} w \curvearrowright S \overset{*}{\Rightarrow}_{\mathcal{G}} w \curvearrowright w \in L(\mathcal{G}')$, weil Ableitung in $\mathcal{G}'$ mit $\alpha_i \overset{*}{\Rightarrow}$ aufgefüllt werden kann zur Ableitung in $\mathcal{G}$

   (b) $w \in L(\mathcal{G})$
   Wir zeigen mit Induktion über der Ableitungslänge v, daß gilt:
   $A \overset{r}{\Rightarrow}_{\mathcal{G}} w \in \Sigma^+ \curvearrowright A \overset{*}{\Rightarrow}_{\mathcal{G}} w$
   $\underline{r = 1:}\ A \Rightarrow w, w \neq \epsilon \curvearrowright A \to w \in P \curvearrowright A \to w\ in P' \curvearrowright A \overset{*}{\Rightarrow}_{\mathcal{G}'} w$
   $\underline{r \to r+1:}\ A \Rightarrow X_1 ... X_k \overset{r}{\Rightarrow}_{\mathcal{G}} w.$
   Dann existiert eine Ableitung $X_i \overset{*}{\Rightarrow}_{\mathcal{G}} w_i$ mit $w = w_1 ... w_k, r_i \leq r$
   Falls $w_i \neq \epsilon, X_i \overset{*}{\Rightarrow}_{\mathcal{G}'} w_i$ nach Ind. Vor.
   Falls $w_i = \epsilon$, so $X_i \in N_\epsilon$. Durch entsprechendes Löschen entsteht:
   $A \Rightarrow_{\mathcal{G}'} X'_1 ... X'_k \overset{*}{\Rightarrow} w'_1 ... w'_j = w$ ♢

Elimination von Kettenregeln

Def.: Eine Regel der Form $A \to B$ heißt *Kettenregel.*

Satz: Jedes $\mathcal{G} =< N, \Sigma, P, S > \in CFG$ läßt sich in eine äquivalente *ϵ-freie* Grammatik $\mathcal{G}' \in < N', \Sigma, P', S > \in CFG$ *ohne Kettenregeln* transformieren.

Beweis: O.B.d.A. Sei $\mathcal{G}$ ϵ-frei. Wir definieren $\mathcal{G}' =< N, \Sigma, P', S >$ durch:
$A \to \alpha \in P' :\curvearrowright \alpha \notin N$ und es gibt $B \in N$ mit $A \in \underline{Pre}^*(B)$ und $B \to \alpha \in P$ ♢

3.3.1 Die Chomsky-Normalform

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

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

Satz: Jedes $\mathcal{G} =< N, \Sigma, P, S > \in CFG$ läßt sich in eine äquivalente Grammatik $\mathcal{G}' \in CNF$ transformieren.

Beweis: O.B.d.A sei $\mathcal{G}$ ϵ-frei und ohne Kettenregeln. Außerdem können wir annehmen, daß für $k \geq 2$ $A \to X_1...X_k \in P \curvearrowright X_i \in N$. Denn $X_i = a$ kann ersetzt werden durch ein neues $C_i \in N$ unter Hinzufügen von $C_i \to a$. Bleibt die Elimination von Regeln der Form $A \to B_1...B_k$ mit $k \geq 3$
Ersetzen durch: $A \to B_1C_1$
$C_1 \to B_2C_2$
$C_{k-2} \to B_{k-1}B_k$
mit neuen N-Symbolen $C_1, ..., C_{k-2}$ ♢

3.3.2 Die Greibach-Normalform, Linksrekursion

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

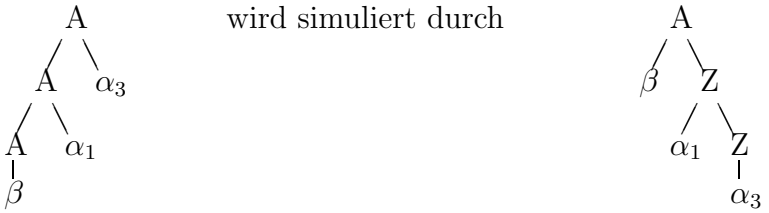
- $A \to aB_1...B_n$ oder
- $A \to a$ oder
- $S \to \epsilon$ und S auf keiner rechten Regelseite.

Bemerkung $\mathcal{G}$ rechtslinear ($A \to wB,\ A \to w$) läßt sich offensichtlich in GNF äquivalent transformieren

Satz: Jedes $\mathcal{G} \in CFG$ läßt sich in ein äquivalentes $\mathcal{G}' \in GNF$ transformieren.

Beweisidee: Elimination linksrekursiver N-Symbole.
Definition: $A \in N$ *linksrekursiv* :$\curvearrowright$ $A \stackrel{+}{\Rightarrow} A\alpha$ für ein $\alpha \in \mathcal{X}^*$.

Spezialfall: Direkte Linksrekursion.
Seien $A \to A\alpha_1 \mid A\alpha_2 \mid ... \mid A\alpha_r$ alle Regeln der Form $A \to A\alpha$
$A \to \beta_1 \mid ... \mid \beta_s$
Transformation:



Neuer Regelsatz: $A \to \beta_1 Z \mid ... \mid \beta_s Z \mid \beta_1 \mid ... \mid \beta_s$
$Z \to \alpha_1 Z \mid ... \mid \alpha_r Z \mid \alpha_1 \mid ... \mid \alpha_r$

3.4 Abschlußeigenschaften von CFL, Pumping-Lemma

Hilfsmittel: Substitutionssatz, Pumping-Lemma

Definition Sei Σ ein Alphabet und für jedes $a \in \Sigma$ Σ_a ein weiteres Alphabet.
Dann heißt

$$\gamma : p(\Sigma^*) \to p((\bigcup_{a\in\Sigma} \Sigma_a)^*)$$

eine *Substitutionsabbildung*, falls:

- $\phi(\{a\}) \subseteq \Sigma_a^*$
- $\phi(\{\epsilon\}) = \{\epsilon\}$
- $\phi(\{a_1...a_r\}) = \phi(\{a_1\}) \cdot \phi(\{a_2\}) \cdot ... \cdot \phi(\{a_r\})$
- $$\phi(L) = \bigcup_{w\in L} \phi(\{w\})$$

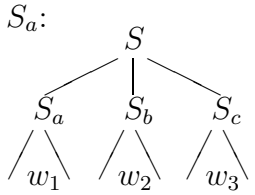
Substitutionssatz: Sei

$$\phi : p(\Sigma^*) \to p((\bigcup_{a\in\Sigma} \Sigma_a)^*)$$

eine Substitutionsabbildung. Dann gilt:

$$L \in CFL(\Sigma), \phi(a) \in CFL(\Sigma_a) \forall a \in \Sigma \quad \curvearrowright \quad \phi(L) \in CFL(\bigcup_{a\in\Sigma} \Sigma_a)$$

Beweis: Kombination kontextfreier Grammatiken mit disjunkten N-Symbol-Mengen. 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 $\phi(a) = L_1$ und $\phi(b) = L_2$. Dann gilt:

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

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

Das Pumping-Lemma (uvwxy-Theorem)

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

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

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

Beweis: O.B.d.A. sei $\mathcal{G} =< N, \Sigma, P, S > \in CNF$ mit $L(\mathcal{G}) = L \backslash \{\epsilon\}$. Sei $n := |N|$, $k := 2^n$ und $z \in L$ mit $|z| \geq k$. Ein Ableitungsbaum $t = \overset{\text{S}}{/_{\text{z}}\backslash}$ hat mindestens 2^n Blätter. Da $\mathcal{G} \in CNF$ 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 zwei Knoten mit gleichen N-Symbolen. Wähle auf p den letzten Knoten, dessen Marke $A \in N$ sich wiederholt. Dann hat sein Teilbaum höchstens $2^n = k$ Blätter.

- $|uwx| \leq 2^n = k$
- $vx \neq \epsilon$, weil $\mathcal{G} \in CNF(A \to BC)$
- $uv^iwx^iy \in L \ \forall i \in \mathbb{N}$ ♢

Lemma: $L = \{a^nb^nc^n | n \geq 1\} \notin CFL(\{a, b, c\})$

Beweis: Währe L kontextfrei, so existiert ein Pumping-Index $k \in \mathbb{N}$. Für $a^kb^kc^k$ existiert eine Zerlegung $uvwxy$ mit den Eigenschaften des Pumpin-Lemmas, insbesondere $uwy \in L$
$a \ldots \underbrace{ab..}_{k}.bc...c = u\underbrace{vwx}_{\leq k}y$
Da $vx \neq \epsilon$, muß $|uwy| \leq 3k$.
Da $|vwx| \leq k$, kann vx nicht gleichzeitig ein a und ein c enthalten. Also muß $uwy = a^k...$ oder $uwy = ...a^k$ sein. ♢

Satz: CFL ist weder unter Durchschnitt noch unter Komplement abgeschlossen.

Beweis: $S \to Sc \mid Ac$ und $S \to aS \mid aA$
$A \to aAb \mid ab$ $A \to bAc \mid bc$
erzeugen die Sprachen $\{a^nb^nc^k \mid n, k \geq 1\}$ bzw. $\{a^kb^nc^n \mid k, n \geq 1\}$ deren Schnitt nicht kontextfrei ist. Da CFL unter Vereinigung abgeschlossen ist, kann Sie nicht unter Komplement abgeschlossen sein, denn $L_1 \cap L_2 = \overline{\overline{L_1} \cap \overline{L_2}}$ ♢

3.5 Entscheidbare Eigenschaften von CFG

Satz: Das Wortproblem und das Leerheitsproblem sind für CFG entscheidbar.

Beweis: $w \in L(\mathcal{G}) \curvearrowright S \in \underline{Pre}^*(\{w\})$
$L(\mathcal{G}) = \emptyset \curvearrowright S \notin \underline{Pre}^*(\Sigma^*)$ ◎ ♢

Bemerkungen:
1.) Das Wortproblem ist in $O(n^3)$ entscheidbar (CYK-Algorithmus).
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 Rechenausdrücken)
  – Prozedurkeller (Speichertechnik für rekursive Prozeduren)

Definition: Seien $Q,\ \Sigma,\ \Gamma$ 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 $\delta : Q \times \Sigma_\epsilon \times \Gamma \to P_{\underline{f}}(endlicheTeilmenge)(Q \times \Gamma^*)$ Transitionsfunktion.
Dann heißt $\mathfrak{A} =< Q, \Sigma, \Gamma, \delta, q_0, F, Z_0 >$ ein Kellerautomat über Σ.
Bezeichnung: $\mathfrak{A} \in PDA(\Sigma)$

Semantik

- Konfigurationsmenge: $Q \times \Sigma^* \times \Gamma^*$
- Einzelschrittrelation: $(q, w, \alpha) \vdash (q', w', \alpha')$
  - Σ-Schritt: $(q, aw, Z_\alpha) \vdash (q', w, \beta\alpha)$ falls $\delta(q, a, Z) \ni (q', \beta)$
  - ϵ-Schritt: $(q, w, Z_\alpha) \vdash (q', w, \beta)$ falls $\delta(q, \epsilon, Z) \ni (q', \beta)$

Die von $\mathfrak{A}$ erkannte Sprache:
$L(\mathfrak{A}) := \{w \in \Sigma^* \mid q_0, w, Z_0) \vdash^* (q, \epsilon, \alpha), q \in F\}$

Bemerkung: Alternative Erkennung mit leerem Keller äquivalent.
$\mathcal{L}(PDA, F) = \mathcal{L}(PDA, \epsilon)$

Definiere $\mathfrak{A}_\mathcal{G} =< Q, \Sigma, \Gamma, \delta, q_0, F, Z_0 > \in PDA(\Sigma)$ durch $Q := \{q\}, \Gamma := N \cup \Sigma, Z_0 := S$

$\delta(q, a, a) := \{(q, \epsilon)\} \quad \forall a \in \Sigma$

$\delta(q, \epsilon, A) := \{(q, \beta) \mid A \to \beta \; in \; P\} \quad \forall A \in N$

Satz: $CFL(\Sigma) \subseteq \mathcal{L}(\Sigma, PDA)$

Beweis: "Simulation von Linksableitungen auf Keller"

Sei $\mathcal{G} =< N, \Sigma, P, S > \in CFG$

Ziel: Nachweis von $L(\mathcal{G}) = L(\mathfrak{A}_\mathcal{G}, \epsilon)$

$A \overset{*}{\Rightarrow} w \quad \curvearrowright (q, wv, A\alpha) \vdash^* (q, v, \alpha) \; \forall v \in \Sigma^*$ und $\alpha \in \Gamma^*$

- "$\curvearrowright$" Induktion über die Ableitungsmenge n
  - $n = 1$: $A \to w \quad (q, wv, A\alpha) \vdash (q, wv, w\alpha) \vdash^* (q, v, \alpha)$
  - $n \to n+1$: $A \Rightarrow v_0 A_1 v_1 ... A_r v_r \overset{n}{\Rightarrow} w$. Dann muß $A_i \overset{n_i}{\Rightarrow} w_i, w = v_0 w_1 v_1 ... w_r v_r$, so daß nach Induktion
    $(q, \underbrace{v_0 w_1 v_1 ... w_r v_r}_{w} v, A\alpha)$
    $\vdash v_0 w_1 ... v_r v, v_0 A_1 v_1 ... A_r v_r \alpha)$
    $\vdash^* (q, w_1 v_1 ... v_r v, A_1 v_1 ... A_r v r \alpha)$
    $\vdash^*_{I.V.} (q, v_1 w_1 ... v_r v, v_1 ... \alpha) ...$
    $\vdash^* (q, v, \alpha)$
- "$\curvearrowleft$" Induktion über die Länge der Berechnung.
  - $n = 1$: Dann muß $A \to \epsilon$ und $w = \epsilon$ gelten, also $A \overset{*}{\Rightarrow} w$
  - $n \to n+1$: $(q, wv, A\alpha) \vdash^{n+1} (q, v, \alpha)$ zerfällt in
    $(q, wv, A\alpha)$
    $\vdash (q, wv, v_0 A_1 v_1 ... A_r v_r \alpha)$
    $\vdash^* (q, w_1 v_1 ... w_r v_r v, A_1 v_1 ... A_r v_r \alpha)$
    $\vdash^{n_1} (q, v_1, w_r v_r v, v_1 ... v A_r v_r \alpha)$
    ...
    $\vdash^{n_r} (q, v_r v, v_r \alpha)$
    $\vdash^* (q, v, \alpha)$

  Nach Induktionsvoraussetzung gibt es Ableitungen
  $A_1 \overset{*}{\Rightarrow} w_1 ... A_r \overset{*}{\Rightarrow} w_r$ so daß $A \Rightarrow v_0 A_1 v_1 ... A_r v_r \overset{*}{\Rightarrow} v_0 w_1 ... v_r = w$ $\diamondsuit$

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

Korollar: CFL ist unter Schnitt mit regulären Sprachen abgeschlossen.

Beweis: Für $L \in CFL(\Sigma)$ und $R \in REG(\Sigma)$ ex.$\mathfrak{A}_L \in PDA(\Sigma)$ 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 Parallelkonstruktion.
$Q := Q_L \times Q_R$
$\delta((q_1, q_2), a, Z) \ni (q_1', q_2', \alpha)$
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: $L = \{ww \mid w \in \{a, b\}^+\} \in CFL$

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 ex. Pumping-Index k mit $a^k b^k a^k b^k = uvwxy$ mit $|vwx \leq k$ und $vx \neq \epsilon$, also auch $uwy \in L'$. Dies ist ein Widerspruch, weil die Zahl der vorderen und hinteren a's bzw. b's nicht mehr gleich ist.
$L' = L \cap \underbrace{\{a\}^+\{b\}^+\{a\}^+\{b\}^+}_{regulär}$. Wäre $L \in CFL$, so auch L'. ♢

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

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

3.6.1 Deterministische Kellerautomaten

Ziel: $\mathcal{L}(\Sigma, DPDA)$ unter Komplement abgeschlossen, somit $\mathcal{L}(\Sigma, DPDA) \subsetneq \mathcal{L}(\Sigma, PDA) = CFL$, d.h. i.A. 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 $(q, a, Z) \in Q \times \Sigma_\epsilon \times \Gamma$
2. $|\delta(q, \epsilon, Z)| = 1 \curvearrowright |\delta(q, a, Z)| = 0$ für alle $a \in \Sigma$

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

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

Satz: $\mathcal{L}(\Sigma, DPDA)$ 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 $\alpha \in \Gamma^*$ mit $(\tilde{q_0}, w, \tilde{Z_0}) \vdash^* (\tilde{q}, \epsilon, \alpha)$.
Dazu müssen Schleifenkonfigurationen eliminiert werden.

Definition: $q, w, \alpha) \in Q \times \Sigma^* \times \Gamma^*$ heißt *Schleifenkonfiguration*, falls für jedes $i \in \mathbb{N}$ ein $q_i \in Q$ und ein $\alpha_i \in \Gamma^*$ ex., so daß $|\alpha_i| \geq |\alpha|$ und $(q, w, \alpha) \vdash^i (q_i, w, \alpha_i)$.

Diese Eigenschaft ist entscheidbar $\curvearrowright$ 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. sei $\mathcal{G} \in CNF$ (auf beliebige CFG übertragbar). Für $\mathcal{G} =< N, \Sigma, P, S > \in CNF$ $(* \; A \to BC, A \to a \; *)$ und $w = a_1 a_2 ... a_n$ mit $n > 0$ definieren wir:
$w_{ij} := a_i a_{i+1} ... a_j$ für $1 \leq i \leq j \leq n$
$N_{ij} := \{A \in N \mid A \stackrel{*}{\Rightarrow} w_{ij}\}$
so daß gilt: $w \in L(\mathcal{G}) \curvearrowright S \in N_{1n}$

Bestimme N_{1n}:

Bestimme	N_{11}	N_{22}	N_{33} ...	N_{nn}
dann	N_{12}	N_{23}	...	$N_{(n-1)n}$
	N_{13}	...	$N_{(n-2)n}$	
	$\vdots$	$\ddots$		
bis	N_{1n}			

mit Hilfe von:

1. $A \in N_{ii} \curvearrowright A \to a_i \in P$
2. $A \in N_{ij}$ mit $i < j \curvearrowright \exists k : i \leq k \leq j \; \exists A \to BC \in P, \; B \in N_{ik}, \; C \in N_{k+1,j}$

Bsp.: $\mathcal{G}$: $S \to AB \mid a$
$A \to BA \mid a$
$B \to AB \mid b$

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

also $baab \in L(\mathcal{G})$

Komplexität:

- Zeit: äußere "for i" $c_1 \cdot n$ Schritte
  "for k" $c_2 \cdot d$ Schritte
  innere "for i" $c_2 \cdot (n-d) \cdot d$ Schritte
  "for d" $c_2 \cdot \sum_{d=1}^{n-1}(n-d) \cdot d$ Schritte

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

- Platz: $O(n^2)$

3.8 Erweiterte kontextfreie Grammatiken (EBNF)

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

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

Semantik: P repräsentiert die Regelmenge $\tilde{P}$ mit $A \to \tilde{\alpha} \in \tilde{P} : \; \tilde{\alpha} \in [\![P(A)]\!]$

Beachte: $\tilde{P}$ ist i.A. unendlich. $L(\mathcal{G}) := L(< N, \Sigma, \tilde{P}, S >)$

Satz: $CFL(\Sigma) = L(\Sigma, ECFG)$

Beweisidee:

- "$\subseteq$" nach Definition.

- "$\supseteq$" Transformation von $ECFG$ nach CFG durch Elimination regulärer Ausdrücke:
  - Disjunktion: Regelalternation
  - Stern: neues NT (non-terminal) mit $A \to \alpha A \mid \epsilon$

3.9 Rekursive endliche Automaten (Syntaxdiagramme)

Syntaxdiagramme entsprechen endlichen Automaten, die sich rekursiv aufrufen können.
Neben $\cdot_q \xrightarrow{a} \cdot_{q'}$ ist auch $\cdot_q \xrightarrow{r} \cdot_{q'}$ möglich.

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

Semantik: Konfigurationen: $Q \times \Sigma^*$
Transitionsrelation: $\vdash \subseteq (Q \times \Sigma^*)^2$ ist definiert durch:

$$\frac{p \in \delta(q,a)}{(q,aw) \vdash (p,w)} \qquad \frac{p \in \delta(q,\epsilon)}{(q,w) \vdash (p,w)}$$

$$\frac{p \in \delta(q,r) \quad (r,w) \vdash^* (s,v) \quad s \in F}{(q,w) \vdash (p,v)}$$

$L(Q) := \{w \in \Sigma^* | q_0, w) \vdash^* (p, \epsilon), p \in F\}$

Satz: $L(\Sigma, RFA) = L(\Sigma, PDA)$

Beweis:

- "$\subseteq$" $\mathfrak{A} =< Q, \Sigma, \delta, q_0, F >\in RFA(\Sigma)$
  Simulation von $\mathfrak{A}$ durch $\mathfrak{A}' =< Q, \Sigma, Q, \delta', q_0, q_0, \emptyset >\in PDA$

  Idee: Keller für "Rücksprungadressen" (Zustände)

  $$\begin{aligned}\delta'(q,a,s) &= \{(p,s) | p \in \delta(q,a)\}\\ \delta'(q,\epsilon,s) &= \{(p,s) | p \in \delta(q,\epsilon)\}\\ &\quad \cup \{(r,ps) | p \in \delta(q,r)\} \text{ "Aufruf"}\\ &\quad \cup \{(s,\epsilon) | q \in F\} \qquad \text{"Rücksprung"}\end{aligned}$$

  für alle $q, s, r, p \in Q$ und $a \in \Sigma$

- "$\supseteq$"
  Für $L \in \mathcal{L}(\Sigma, PDA)$ existiert $\mathcal{G} \in< N, \Sigma, P, S >\in CNF$ mit $L = L(\mathcal{G})$. Konstruiere
  $\mathfrak{A} =< Q, \Sigma, \delta, q_0, F >\in RFA$ durch $Q := N \dot\cup \{q_f\}, q_0 := S, F := \{q_f\}$ und
  $c \in \delta(A, B)$, falls $A \to BC$
  $q_f \in \delta(A, a)$, falls $A \to a$
  $q_f \in \delta(S, \epsilon)$, falls $S \to \epsilon$ $\diamondsuit$

Kapitel 4

Turingmaschinen und Aufzählbare Sprachen

4.1 Chomsky-Grammatiken

Verallgemeinerung kontextfreier Grammatiken, kontextabhängige Ersetzung, Wertersetzung.

Definition: Seien $N, \Sigma, \mathcal{X}$ und S wie bei CFG. Sei ferner $P \subseteq \mathcal{X}^*\{N\}\mathcal{X}^* \times \mathcal{X}$, P endlich.
Dann heißt $\mathcal{G} =< N, \Sigma, P, S >$ eine *Chomsky-Grammatik*

Semantik: $\pi = \alpha_1 \to \alpha_2 \in P$ bestimmt die Ableitungsrelation $\Rightarrow_\pi \subseteq \mathcal{X}^* \times \mathcal{X}^*$ durch $\beta_1 \Rightarrow_\pi \beta_2$ $\curvearrowleft$ es ex. $\gamma, \delta \in \mathcal{X}^*$ mit $\beta_1 = \gamma\alpha_1\delta$ und $\beta_2 = \gamma\alpha_2\delta$
Ableitungsrelation von $\mathcal{G}$:

$$\Rightarrow_\mathcal{G} := \bigcup_{\pi \in P} \Rightarrow_\pi$$

$\mathcal{G}$ erzeugt $L(\mathcal{G}) := \{w \in \Sigma^* \mid S \Rightarrow_\mathcal{G}^* w\}$

Folien 4.2 und 4.3
Abschlußeigengschaften von $\mathcal{L}_0$ und $\mathcal{L}_1$.

Definition: Sei $\mathcal{G} =< N, \Sigma, P, S >$ vom Typ-0. $\mathcal{G}$ heißt normiert, wenn gilt:

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

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

Satz: Zu jedem $\mathcal{G} =< N, \Sigma, P, S >$ vom Typ-0 läßt sich eine äquivalente normierte Grammatik $\mathcal{G}' =< N', \Sigma, P', S' >$ konstruieren.

Beweis:

1. Terminalsymbolbedingung: $a \in \Sigma \rightarrow A_a$ neues N-Symbol a durch Aa ersetzen (in P), $Aa \rightarrow a$ hinzufügen.

2. Simulation von $A_1...A_n \rightarrow B_1...B_m (n \geq 1, m \in \mathbb{N})$ durch folgende Regeln:

$$\begin{array}{lcl} A_1A_2A_3...A_{n-1}A_n & \rightarrow & A'_1A_2A_3...A_{n-1}A_n \\ A'_1A_2A_3...A_{n-1}A_n & \rightarrow & A'_1A'_2A_3...A_{n-1}A_n \end{array}$$

$$\begin{array}{lcl} A'_1A'_2A'_3...A'_{n-1}A_n & \rightarrow & A'_1A'_2A'_3...A'_{n-1}A'_n \\ A'_1A'_2A'_3...A'_{n-1}A'_n & \rightarrow & B_1A'_2A'_3...A'_{n-1}A'_n \end{array}$$

2 Fälle:

Fall 1: $n \leq m$ $\quad B_1...B_{n-1}A'_n \rightarrow B_1...B_m$

Fall 2: $n > m$ $\quad B_1...B_mA'_{m+1}A'_n \rightarrow B_1...B_mA'_{m+1}...A'_n$

$$B_1...B_mA'_n \rightarrow B_1...B_m$$

A'_i neue N-Symbole: Sie verhindern Seiteneffekte: neue Satzformen, mehr Ableitungen

$\diamondsuit$

Satz: $\mathcal{L}_0(\Sigma)$ ist unter Substitution abgeschlossen, d.h.: ist

$$\phi : \mathcal{P}(\Sigma^*) \rightarrow P(\bigcup_{a \in \Sigma} \Sigma_a)^*$$

eine Substitutionsabbildung, so gilt:

$$L \in \mathcal{L}_0(\Sigma),\ \phi(a) \in \mathcal{L}_0(\Sigma_a) \forall a \in \Sigma \quad \curvearrowright \quad \phi(L) \in L_0(\bigcup_{a \in \Sigma} \Sigma_a)$$

Beweis: Konstruktion von kontextfreien Grammatiken (CFG) nicht anwendbar, wegen möglicher neuer Satzformen.
Idee: Sequentialisierung mit einem Kontrollsymbol (Siehe Bild auf Blatt 26, Rückseite)
$L = L(\mathcal{G})$ mit $\mathcal{G} =< N, \Sigma, P, S >$ vom Typ-0. $\phi(a) = L_a = L(\mathcal{G}_a)$ mit $\mathcal{G}_a =< N_a, \Sigma_a, P_a, S_a >$ vom Typ-0. O.B.d.A.: $\mathcal{G}, \mathcal{G}_a$ normiert, N, N_a disjunkt. Konstruktion von $\overline{\mathcal{G}}$ mit $L(\overline{\mathcal{G}}) = \phi(L)$.

$$\overline{N} := N \dot{\cup} (\dot{\bigcup_{a \in \Sigma}} N_a) \dot{\cup} \{A_a \mid a \in \Sigma\} \dot{\cup} \{\overline{S}, C\}$$

$$\overline{\Sigma} := \bigcup_{a\in\Sigma} \Sigma_a$$

$$\overline{P} := P_0 \cup P_1 \cup P_2 \cup (\bigcup_{a\in\Sigma} P_a) \cup \{\overline{S}_R \to CS, C \to \epsilon\}$$

mit P_0 entstehe aus P durch Ersetzen von a durch $A_a \quad (a \in \Sigma)$

$P_1 := \{CA_a \to CSa \mid a \in \Sigma\}$

$P_2 := \{Ca \to aC \mid a \in \overline{\Sigma}\}$

1. $\phi(L) \subseteq L(\overline{\mathcal{G}})$:

$$\begin{aligned}\overline{S} \Rightarrow CS \overset{*}{\Rightarrow} CAa_1...Aa_r \overset{*}{\Rightarrow} & CS_{a_1}A_{a_2}...A_{a_r}\\ & Cw_1A_{a_2}...A_{a_r}\\ & w_1CA_{a_2}...\\ & ...\\ & w_1...w_r\end{aligned}$$

2. $L(\overline{\mathcal{G}}) \subseteq \phi(L)$:
   keine Ableitungen neuer Wörter, weil $\mathcal{G}$ normiert ist und daher A_a auf keiner linken Regelseite, weil die N-Alphabete disjunkt sind und weil C die Teilableitungen der $\mathcal{G}_a$ sequentialisiert. ♢

Korollar: $\mathcal{L}_0(\Sigma)$ ist unter den regulären Operationen abgeschlossen.

$\underbrace{L}_{\{a\}}, \underbrace{L'}_{\{b\}} \in \mathcal{L}_0(\Sigma) \quad \curvearrowright \quad \underbrace{L \cup L'}_{\{a,b\}}, \underbrace{LL'}_{\{ab\}}, \underbrace{L^*}_{\{a\}^*} \in \mathcal{L}_0(\Sigma).$

Beweis: Wie für CFL.

Satz: $\mathcal{L}_0(\Sigma)$ ist unter Schnitt abgeschlossen.

Beweis: $L_i = L(\mathcal{G}_i)$ und $\mathcal{G}_i =< N_i, \Sigma, P_i, S_i >$ vom Typ-0 für $i = 1, 2$.
O.B.d.A. sei $N_1 \cap N_2 \neq \emptyset$
Konstruktion von $\mathcal{G} =< N, \Sigma, P, S >$:
$N := N_1 \cup N_2 \cup \{A_a \mid a \in \Sigma\} \dot\cup \{S, C_1, C_2\}$
$P := P_1 \cup P_2 \cup Q$
$Q := \{S \to C_1S_1C_2S_2C_1,$
$\quad C_2a \to A_aC_2,$
$\quad bA_a \to A_ab,$
$\quad C_1A_aa \to aC_1,$
$\quad C_1C_2C_1 \to \epsilon\}$ ♢

4.1.1 Kontextrekursive Grammatiken und Sprachen

Typ-1 $\sim$ Platzbedarf von Berechnungen.

Definition: $\mathcal{G} =< N, \Sigma, P, S >$ heißt von wachsender Länge $:\curvearrowright$ für jede Regel $\alpha \to \beta \in P$ gilt $|\alpha| \leq |\beta|$, es sei denn, daß $S \to \epsilon \in P$ und S auf keiner rechten Regelseite.

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

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

Beweis: Normierung

Definition: (Platzbedarf)
Sei $\mathcal{G} =< N, \Sigma, P, S >$ von Typ-0, $\delta = (S \Rightarrow \alpha_0 \Rightarrow \alpha_1 \Rightarrow ...\alpha_n)$ eine Ableitung von $\mathcal{G}$ und $w \in \Sigma^*$. Dann heißt:
$\underline{pl}_{\mathcal{G}}(\delta) := \underline{max}\{|\alpha_i| \mid 0 \leq i \leq n\}$ der Platzbedarf von δ und
$\underline{pl}_{\mathcal{G}}(w) := \underline{min}\{\underline{pl}_{\mathcal{G}}(\delta) \mid \delta = (S \Rightarrow ... \Rightarrow w)\}$

Folgerungen:

1. $\underline{pl}_{\mathcal{G}}(w) \geq |w|$
2. $\mathcal{G}$ vom Typ-2, $w \notin \epsilon \curvearrowright \underline{pl}_{\mathcal{G}}(w) = |w|$

Platzbedarfssatz:
Sei $\mathcal{G}$ vom Typ-0, $p \in \mathbb{N},\ p > 0$, so daß für alle $w \in L(\mathcal{G}) \setminus \{\epsilon\} : \underline{pl}_{\mathcal{G}}(w) \leq p|w|$
Dann ist $L(\mathcal{G}) \in L_1$

Beweis: Sei $p = 1$, also für $w \in L(\mathcal{G}) \setminus \epsilon\ :\ \underline{pl}_{\mathcal{G}}(w) = |w|$
Elimination verkürzender Regeln durch auffüllen. Konstruktion einer äquivalenten Grammatik von wachsender Länge: $\mathcal{G}' =< N \dot{\cup} \{\$\}, \Sigma, P', S >$

1. $\alpha \to \beta \in P$ mit $|\alpha| = |\beta| + i\ (i > 0) \curvearrowright \alpha \to \$^i\beta \in P'$
2. $\alpha \to \beta \in P$ mit $|\alpha| \leq |\beta| \curvearrowright \$^i\alpha \to \beta \in P'\ \forall j = 0, 1, ..., |\beta| - |\alpha|$
3. $\$X \to X\$, X\$ \to \$X \in P'\ \forall X \in \mathcal{X}$

Wegen $\underline{pl}_{\mathcal{G}}(w) = |w|$ lassen sich die eingeschlossene \$ löschen.
$p > 1$: Induktion. Idee: $N' \supseteq N^P$
Abschlußeigenschaften von $\mathcal{L}_1(\Sigma)$ mit Hilfe des Platzbedarfssatzes. $\diamondsuit$

Satz: $\mathcal{L}_1(\Sigma)$ ist unter ϵ-freien Substitutionen ($\epsilon \notin \phi(a)$) abgeschlossen.

Beweis: Platzbedarf der Ausgangsgrammatik mit $p = 1 \curvearrowright$ Platzbedarf der Substitutionsgrammatik $|w| + 1 \leq 2|w|$ wegen zusätzlicher Kontrollsymbole C und fehlender ϵ-Regeln.

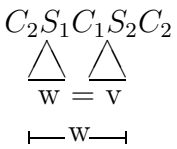
$\diamondsuit$

Korollar: $\mathcal{L}_1(\Sigma)$ ist unter regulären Operationen $(\cup, \cdot, *)$ abgeschlossen.

Beweis: Reduktion auf ϵ-freie Sprachen: $L \in L_1(\Sigma) \curvearrowright L \setminus \{\epsilon\} \in L_1(\Sigma)$ Für ϵ-freie $L, L' \in L_1(\Sigma)$ folgt $L \cup L', LL', L^* \in L_1(\Sigma)$ wie für Typ-0-Sprachen durch ϵ-freie Substitution. Falls $\epsilon \in L$ und/oder $\epsilon \in L'$, so folgert man aus den ϵ-freien Teilsprachen,
z.B.: $(L \dot{\cup} \{\epsilon\})(L' \dot{\cup} \{\epsilon\}) = LL' \cup L \cup L' \cup \{\epsilon\} \in L_1(\Sigma)$ $\diamondsuit$

Korollar: $\mathcal{L}_1(\Sigma)$ ist unter Durchschnitt abgeschlossen.

Beweis: $L_1, L_2 \in \mathcal{L}_1(\epsilon)$ sind mit linearem Platzbedarf $(p = 1)$ erzeugbar. Die $\cap$-Konstruktion für Typ-0:



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

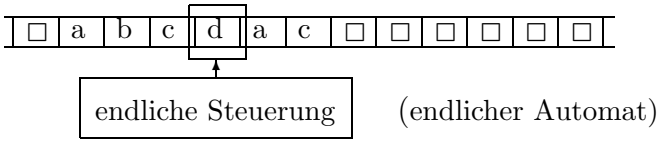
Korollar: $\mathcal{L}_2(\Sigma) \subsetneqq \mathcal{L}_1(\Sigma)$

Beweis: $\mathcal{L}_2(\Sigma)$ ist nicht unter $\cap$ abgeschlossen. $\diamondsuit$

Ein lange offenes Problem: Abschluß $\mathcal{L}_1$ unter Komplement:
1987: $\mathcal{L}_1$ ist unter Komplement abgeschlossen.

4.2 Turingmaschinen, linear beschränkte Automaten

Ziel: $\mathcal{L}_0(\Sigma) = \mathcal{L}(\Sigma, TM)$
$\mathcal{L}_1(\Sigma) = \mathcal{L}(\Sigma, LBA)$
(unendliches Band, endlicher Anteil beschrieben)



Definition: (Nicht-deterministische erkennende Turingmaschinen)
Seien die folgenden nicht-leeren, endliche Mengen gegeben:

- Q Zustandsmenge
- Σ Eingabealphabet
- Γ Arbeitsalphabet mit $\Sigma \subseteq \Gamma$

Ferner:

- $q_0 \in Q$ Anfangszustand
- $\square \in \Gamma \setminus \Sigma$ Blank
- $F \subseteq Q$ Endzustandsmenge

und die Transitionsfunktion:

- $\delta : Q \times \Gamma \to p(Q \times \Gamma \times \{L, N, R\})$

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

Semantik:
Konfigurationsmenge: $Q \times \Gamma^* \times \Gamma \times \Gamma^* =: \underline{Conf}(\mathfrak{A})$
Einzelschrittrelation: $\vdash_{\mathfrak{A}} \subseteq Conf(\mathfrak{A})^2$ 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 $\delta(q, X) \ni (q', X', L)$
- $(q, \epsilon, X, \beta) \vdash (q', \epsilon, \square, X'\beta)$ falls $\delta(q, X) \ni (q', X', L)$
- $(q, \alpha, X, Y\beta) \vdash (q', \alpha X', Y, \beta)$ falls $\delta(q, X) \ni (q', X', R)$
- $(q, \alpha, X, \epsilon) \vdash (q', \alpha X', \square, \epsilon)$ falls $\delta(q, X) \ni (q', X', R)$

Anfangskonfiguration von $w \in \Sigma^*$:

$$\kappa(w) := \begin{cases} (q_0, \epsilon, a, v) \text{ falls } w = av \\ (q_0, \epsilon, \square, \epsilon) \text{ falls } w = \epsilon \end{cases}$$

Endkonfiguration (q, α, X, β) falls $q \in F$
Die von $\mathfrak{A}$ erkannte Sprache:
$L(\mathfrak{A}) := \{w \in \Sigma^* \mid \kappa(w) \vdash^* (q, \alpha, X, \beta), q \in F\}$

Beachte: Erreichen von $q \in F$ genügt, kein Anhalten gefordert. Die Klasse der TM-erkennbaren Sprachen: $\mathcal{L}(\Sigma, TM)$

Satz: $\mathcal{L}(\Sigma, TM) = \mathcal{L}_0(\Sigma)$

Beweis:

1. $\mathcal{L}(\Sigma, TM) \subseteq \mathcal{L}_0(\Sigma)$
   $\mathfrak{A} =< Q, \Sigma, \Gamma, q_0, \square, F, \delta > \in TM(\Sigma)$
   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 $a_1...a_r \in \Sigma^*,\ m, n \in \mathbb{N}$ erzeugt man:
   (*) $(\Sigma, \square)^m q_0(a_1, a_1)(a_2, a_2)...(a_r, a_r)(\epsilon, \square)^n$
   Dazu $G =< N, \Sigma, P, S >$ mit
   $N := Q \dot\cup (\Sigma_\epsilon \times \Gamma) \dot\cup \{S, C_1, C_2\}$
   $P : S \to (\epsilon, \square)S \mid q_0 C_1$
   $C_1 \to (a, a)C_1 \mid C_2 \quad (a \in \Sigma)$
   $C_2 \to (\epsilon, \square)C_2 \mid \epsilon$
   (P: erzeugen von (*))
   $q(a, X) \to q'(a, X')$ falls $\delta(q, X) \ni (q', X', N)$
   $q(a, X) \to (a, X')q'$ falls $\delta(q, X) \ni (q', X', R)$
   $(b, Y)q(a, X) \to q'(b, Y), (a, X')$ falls $\delta(q, X) \ni (q', X', L)$
   $q \to \epsilon$ falls $q \in F$ löschen von q und der 2. Spur
   $(a, X) \to a\ (a \in \Sigma_\epsilon)$ ursprüngliches Wort bleibt übrig.
   Es folgt: $L(\mathcal{G}) = L(\mathfrak{A})$

2. $\mathcal{L}_0(\Sigma) \subseteq \mathcal{L}(\Sigma, TM)$ Simulation einer Typ-0-Sprache durch TM. Neben Eingabewort Ableitung von $\mathcal{G}$ simulieren und erzeugtes Wort mit der Eingabe vergleichen. Technisch aufwendig: Einschieben und Löschen von Worten auf einem Turingband. $\diamondsuit$

Automatencharakterisierung von $\mathcal{L}_1(\Sigma)$ durch Platzbedarf
Definition: (Platzbedarf für TM)
Sei $\mathfrak{A} \in TM(\Sigma), \gamma = (\kappa_0 \vdash \kappa_1 \vdash ... \vdash \kappa_n)$ eine Berechung und $w \in L(\mathfrak{A})$. Dann ist:

- $|\kappa_i| = |\alpha X \beta|$ falls $\kappa_i = (q, \alpha, X, \beta)$
- $bv_{\mathfrak{A}}(\gamma) := \underline{max}\{|\kappa_i| \mid 0 \le i \le n\}$
- $bv_{\mathfrak{A}}(w) := \underline{min}\{bv_{\mathfrak{A}} | \gamma \text{ erkennt } w\}$

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

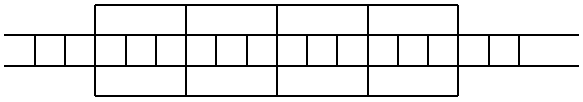
Definition: $\mathfrak{A} \in TM$ heißt *linear beschränkt*, wenn $p \geq 1$ existiert, so daß für alle $w \in L(\mathfrak{A}) \setminus \{\epsilon\} \quad bv_{\mathfrak{A}}(w) \leq p \cdot |w|$ gilt.

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

Definition: $\mathfrak{A} \in LBA(\Sigma) \ :\curvearrowright \ \mathfrak{A}\ lbTM$ mit $p = 1$, d.h. nur Eingabefelder werden benutzt.

Satz: $\mathcal{L}_1(\Sigma) = \mathcal{L}(\Sigma, LBA)$

Beweisidee: Kompression des benutzten Bandbereichs durch Vergrößerung des Arbeitsalphabets $\Gamma^L := \Gamma^P$



4.2.1 Deterministische Turingmaschine, k-Band TM

Reduktion nicht-det. TM auf det. TM in 2 Schritten:

1. Simulation einer ndTM durch 3-Band-TM
2. Reduktion von k-Band-dTM auf 1-Band dTM

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

Konfigurationen: $Q \times (\Gamma^* \times \Gamma \times \Gamma^*)^k$

Einzelschrittrelation: simultanes Arbeiten auf k-Bändern gemäß δ

Anfangskonfiguration für $w \in \Sigma^*$:

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

Satz: Zu jedem $\mathfrak{A} \in TM$ läßt sich ein äquivalenter $\mathfrak{A}' \in 3 - dTM$ konstruieren.

Beweis: Für $\delta_{\mathfrak{A}} : Q \times \Gamma \rightarrow p_f(Q \times \Gamma \times \{L, N, R\})$ gelte:

$max\{|\delta(q,X)| \mid (q,X) \in Q \times \Gamma\} =: r$
Bezeichnung der Alternativen von $\mathfrak{A}$ durch Elemente von $[r] := \{1,...,r\}$
Berechnung von $\mathfrak{A}$ bestimmt $\zeta \in [r]^*$
Umgekehrt Auswahl einer Berechnung durch ein ζ als Steuerwert.

- Band 1: Eingabe, wiederholtes Lesen.
- Band 2: Erzeugung der Zahlenfolgen $\zeta \in [r]^*$
- Band 3: Simulation von Q 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. $\diamondsuit$

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

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

Bemerkung: Für LBA's ist diese Frage offen: das LBA-Problem.

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 ein effektives Verfahren gibt, welches für jedes $w \in \Sigma^*$ feststellt, ob $w \in L$ oder ob $w \notin L$ gilt.

Folgerung: Für $L \subseteq \Sigma^*$ gilt: L entscheidbar $\curvearrowright$ L und $\Sigma^* \setminus L$ aufzählbar.
(Reißverschlußverfahren)

Formalisierung: TM mit Ausgabe.

Satz: $\{L \subseteq \Sigma^* \mid L \text{ aufzählbar } \} = \mathcal{L}(\Sigma, dTM)$

Church-Turing-These: $\curvearrowright$ Jede effektiv beschreibbare Sprache ist eine Typ-0-Sprache.

Satz: $\mathcal{L}_0(\Sigma) \subsetneqq p(\Sigma^*)$

Beweis: Die Menge der Typ-0-Grammatiken über Σ läßt sich abzählen (nach ihrer Größe).

Also ist $\mathcal{L}_0(\Sigma)$ abzählbar. $p(\Sigma^*)$ ist jedoch überabzählbar.

Diagonalverfahren von Cantor:
$|\Sigma| = 1, \Sigma^* = \mathbb{N}, L \subseteq \Sigma^* \mapsto \underline{char}_L : \mathbb{N} \to \{0,1\}$
Angenommen: $p(\mathbb{N})$ sei abzählbar.

	0	1	2	3	
0	0	1	1	0	
1	1	0	1	0	
2			$\ddots$		
3					

$f : \mathbb{N}^R \to \{0,1\}$
$f(i,j) = 1 \quad \curvearrowright \underline{char}_{L-i}(d) = 1$ ♢

Definition: $L_{diag} \subseteq \mathbb{N}$ durch $\underline{char}_{L_{diag}}(j) := 1 - f(j,j)$, dann kann L_{diag} nicht in der Abzählung vorkommen.

Satz: $DEC(\Sigma) := \{L \in \Sigma^* \mid L \text{ entscheidbar}\} \subsetneq \mathcal{L}_0(\Sigma)$

Beweis: Die Sprache Univ ist nicht entscheidbar. Diagonalisierungsschluß (Selbstanwendungsproblem). Vgl. Vorl. BuK. ♢

Satz: Jede Typ-1-Sprache ist entscheidbar.

Beweis: Sei G eine Typ-1-Grammatik über Σ und $w \in \Sigma^*$

Fall 1: $w = \epsilon \in L \quad \curvearrowright S \to \epsilon \in P$
Fall 2: $w \neq \epsilon \in L \quad \curvearrowright \exists$ Ableitung: $S \Rightarrow \alpha_1 \Rightarrow ... \Rightarrow \alpha_n = w$ mit $|\alpha_i| \leq |w|$
Nur endlich viele solcher Ableitungen. ♢

Satz: Es gibt entscheidbare, nicht-kontextsensitive Sprachen.

Beweis: Literatur.

Kapitel 5

Unentscheidbare Probleme

Ziel: Die Folgenden Probleme sind unentscheidbar:

- das Wortproblem für Typ-0-Beschreibugnen (BuK);
- das Leerheitsproblem für Typ-1-Beschreibungen;
- das Äquivalenzproblem für Typ-2-Beschreibungen.

Bemerkung: Das Äquivalenzproblem für $DPDA$'s wurde kürzlich als entscheidbar nachgewiesen.

Hilfsmittel: Das Postsche Korrespondenzproblem (BuK). Seien $w = (w_1, w_2, ..., w_n)$ und $v = (v_1, v_2, ..., v_n)$ mit $w_i, v_i \in \Sigma^*$ und $1 \leq i \leq n$

Beispiel: Eingabe $(1, 10, 011)$ und $(101, 00, 11)$, Indizes: $(1, 3, 2, 3)$:
1 011 10 011 = 101 11 00 11

Dann heißt $(i_1, ..., i_k) \in \{1, .., n\}^k$ eine *Lösung von* $PCP(w, r)$,falls $w_{i_1} w_{i_2} ... w_{i_k} = v_{i_1} v_{i_2} ... v_{i_k}$

Es gilt: Das Postsche Korrespondenzproblem ist unentscheidbar, d.h. 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 PCP auf das Leerheitsproblem (Typ-1)
Sei $w = (w_1, ..., w_n), v = (v_1, ..., v_n) \in (\Sigma^*)^n$
Konstruktion von $\mathfrak{A}(w, r) \in lbTM$ mit $PCP(w, v)$ lösbar $\curvearrowright L(\mathfrak{A}(w, v)) \in \emptyset$
$\mathfrak{A}(w, v)$ erzeugt bei Eingabe von $u \in \Sigma^+$ alle Folgen $(i_1, ..., i_k) \in \{1, ..., n\}^k$ mit $k \leq |u|$ und prüft jeweils, ob $w_{i_1} ... w_{i_k} = v_{i_1} ... v_{i_k}$. $\mathfrak{A}$ erkennt u, falls eine solche Indexfolge auftritt. Platzbedarf ist linear in $|u|$. Wäre das Leerheitsproblem entscheidbar, so auch PCP.

$\diamondsuit$

Satz: Schnittproblem für Kontextfreie Sprachen.
Es ist nicht entscheidbar, ob für $\mathcal{G}_1, \mathcal{G}_2 \in CFG$ gilt: $L(\mathcal{G}_1) \cap L(\mathcal{G}_2) = \emptyset$

Beweis: Reduktion des PCP auf das Schnittproblem von CFL. Sei $w = (w_1, ..., w_n)$ und $v = (v_1, ..., v_n) \in (\Sigma^*)^n$. Konstruiere $\mathcal{G}_w, \mathcal{G}_v \in CFG$, so daß $PCP(w, v)$ lösbar $\curvearrowright L(\mathcal{G}_w) \cap L(\mathcal{G}_v) \neq \emptyset$
$\tilde{\Sigma} := \Sigma \dot{\cup} \{b_1, ..., b_n\}$
$\mathcal{G}_w :=< \{S_w\}, \tilde{\Sigma}, P_w, S_w >$ mit den Regeln:
$P_w := S_w \to w_i S_w b_i, S_w \to w_i b_i \mid 1 \leq i \leq n\}$
G_v analog.
Es folgt:
$L_w := L(\mathcal{G}_w) = \{w_{i_1} w_{i_2} ... w_{i_k} b_{i_k} b_{i_{k-1}} ... b_{i_1} \mid k \geq 1, 1 \leq i_j \leq n\}$ L_v analog.
Und $L - w \cap L_v \neq \emptyset \curvearrowright \exists I - 1 ... i_k \in 1...n$,
$w_{i_1} ... w_{i_k} b_{i_k} ... b_{i_1} = v_{i_1} ... v_{i_k} b_{i_k} ... b_{i_1}$
also $w_{i_1} ... w_{i_k} = v_{i_1} ... v_{i_k} \curvearrowright PCP(w, v)$ lösbar. $\diamondsuit$

Korollar: Das Schnittproblem für $DPDA$ ist nicht entscheidbar.

Beweis: $\mathcal{G}_w$ und $\mathcal{G}_v$ sind durch deterministische Kellerautomaten simulierbar. $\diamondsuit$

Satz: Es ist nicht entscheidbar, ob für $\mathcal{G}_1, \mathcal{G}_2 \in CFG$ gilt: $L(\mathcal{G}_1) = L(\mathcal{G}_2)$ (Äquivalenzproblem)

Beweis: Reduktion des Schnittproblems für $DPDA$'s auf das Äquivalenzproblem von CFG $\diamondsuit$

Beachte:

1. Zu $\mathfrak{A} \in DPDA(\Sigma)$ läßt sich $\overline{\mathfrak{A}} \in DPDA(\Sigma$ konstruieren mit $L(\overline{\mathfrak{A}} = \overline{L(\mathfrak{A})}$
2. Zu $\mathfrak{A} \in PDA(\Sigma)$ läßt sich $\mathcal{G}_{\mathfrak{A}} \in CFG(\Sigma$ konstruieren mit $L(\mathcal{G}_{\mathfrak{A}} = L(\mathfrak{A})$
3. Zu $\mathcal{G}_1, \mathcal{G}_2 \in CFG(\Sigma)$ läßt sich $\mathcal{G}_v \in CFG(\Sigma)$ konstruieren mit $L(\mathcal{G}_v) = L(\mathcal{G}_1) \vee L(\mathcal{G}_2)$

Sei $\mathfrak{A}_1, \mathfrak{A}_2 \in DPDA$
$L(\mathfrak{A}_1) \cap L(\mathfrak{A}_2) = \emptyset$
$\curvearrowright$ $L(\mathfrak{A}_1) \subseteq L(\overline{\mathfrak{A}_2})$
$\curvearrowright$ $L(G_{\mathfrak{A}_1}) \subseteq L(\mathcal{G}_{\overline{\mathfrak{A}_2}})$
$\curvearrowright$ $L(\mathcal{G}_{\mathfrak{A}_1}) \cup L(\mathcal{G}_{\overline{\mathfrak{A}_2}}) = L(\mathcal{G}_{\overline{\mathfrak{A}_2}})$
$\curvearrowright$ $L(\mathcal{G}_v) L(\mathcal{G}_{\mathfrak{A}_2})$

Die Entscheidbarkeit des Äquivalenzproblems für kontextfreie Grammatiken würde daher die Entscheidbarkeit des Schnittproblems nach sich ziehen: Widerspruch.