RWTH Aachen
Sommersemester 2003

Automatentheorie und Formale Sprachen

Prof. Dr. K. Indermark (Lehrstuhl i2)

Vorlesungsmitschrift von Mark Wiesemann

Stand: 22. September 2003

Hinweise auf evt. Fehler bitte an

fehler@mark-wiesemann.de

Vorwort

Dieses Skript basiert auf meiner Mitschrift der Vorlesung „Automatentheorie und Formale Sprachen“ von Prof. Dr. K. Indermark im Sommersemester 2003 an der RWTH Aachen. Es handelt sich nicht um eine offizielle Veröffentlichung des Informatik-Lehrstuhls i2.

Ich übernehme keine Gewähr für die Fehlerfreiheit und Vollständigkeit des Skripts. Korrekturen können an fehler@mark-wiesemann.de geschickt werden.

Mark Wiesemann, 22. September 2003

Inhaltsverzeichnis

1 Alphabete, Wörter, Sprachen

1.1 Einführung

Zeichenreihen als Grundobjekte der Informatik:

- Präzisierung des Algorithmusbegriffs durch Turing-Maschine
- Kommunikation mit Rechner über Tastatur
- Informationsverarbeitung; Transformation mit Bitstrings

Grundbegriff 1.1 (Alphabet). Σ: Alphabet, nicht-leere endliche Menge
$a \in \Sigma$: Buchstabe, Zeichen, Character, Symbol

Grundbegriff 1.2 (Menge der Wörter über Σ). Σ^*: Menge der Wörter über Σ
$\Sigma^* := \{a_1 a_2 \dots a_n | n \in \mathbb{N}\}$ Wörter, Zeichenreihen, Strings
$n = 0$: das leere Wort, Bezeichung: ε (Metasprache)

Beispiel 1.3 (zu Grundbegriff 1.2). Bitstrings, Webadresse, Dezimalzahlen, RGB, Java-Programme

1.2 Operationen auf Σ^*

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

  (i) $\cdot$ ist assoziativ: $(u \cdot v) \cdot w = u \cdot (v \cdot w)$

  (ii) ε ist $\cdot$-neutral: $\varepsilon \cdot w = w \cdot \varepsilon = w$

  Sprechweise: $\langle \Sigma^*, \_ \cdot \_, \varepsilon \rangle$ ist ein MONOID.
  Bemerkung: freie Erzeugung:

  $$a_1 \dots a_n = b_1 \dots b_m \curvearrowright n = m, a_i = b_i \ (i = 1, \dots, n)$$

- Länge eines Wortes $w = a_1 a_2 \dots a_n \in \Sigma^*$:
  $| a_1 \dots a_n | := n$, falls alle $a_i \in \Sigma^*$
  also: $| \varepsilon | = 0$
  und: $| wv | = | w | + | v |$
- Potenzen eines Wortes $w \in \Sigma^*$:
  $w^0 := \varepsilon$
  $w^{n+1} := w^n w$

- Spiegelbild eines Wortes
  $\varepsilon^R := \varepsilon$
  $(wa)^R := aw^R$

1.3 Formale Sprachen über Σ

Grundbegriff 1.4 (Menge der formalen Sprachen über Σ). $P(\Sigma^*) := \{L | L \subseteq \Sigma^*\}$

Beispiel 1.5 (zu Grundbegriff 1.4). $\emptyset, \{\varepsilon\}, \{w_1, \ldots, w_n\}, \Sigma^*$, Menge der URLs / Java-Programme / HTML-Beschreibungen

1.4 Operationen auf $P(\Sigma^*)$

- boolesche Operationen:
  $L_1 \cup L_2, L_1 \cap L_2, \overline{L} := \Sigma^* \setminus L$ (Mengendifferenz)
- Komplexprodukt:
  $L_1 L_2 := \{wv | w \in L_1, v \in L_2\}$
  $L_1 L_1 := \{wv | w, v \in L_1\}$ („Jeder mit jedem“)
- Potenzen einer Sprache:
  $L^0 := \{\varepsilon\}$
  $L^{n+1} := L^n L$
- Stern einer Sprache (Iteration, Repetition):
  $L^* := \bigcup_{n \in \mathbb{N}} L^n \curvearrowright \emptyset^n = \{\varepsilon\}$
- Spiegelbild einer Sprache:
  $L^R := \{w^R | w \in L\}$
- reguläre Operationen:
  $L_1 \cup L_2, L_1 L_2, L^*$

2 Reguläre Ausdrücke und endliche Automaten

2.1 Reguläre Ausdrücke

Ein regulärer Ausdruck beschreibt eine formale Sprache, die sich mit Hilfe regulärer Operationen (Vereinigung, Komplexprodukt, Stern) aus einfachen Sprachen erzeugen lässt.

Definition 2.1 (Syntax von RegE(Σ)**).** Sei Σ ein Alphabet.
Die Menge RegE(Σ) der REGULÄREN AUSDRÜCKE ÜBER Σ ist induktiv definiert durch:

(i) $\Lambda \in \text{RegE}(\Sigma)$

(ii) $a \in \text{RegE}(\Sigma)$ für jedes $a \in \Sigma$

Wenn $\alpha, \beta \in \text{RegE}(\Sigma)$, so auch:

(iii) $(\alpha \vee \beta) \in \text{RegE}(\Sigma)$ („Alternative“)

(iv) $(\alpha \cdot \beta) \in \text{RegE}(\Sigma)$ („Konkatenation“)

(v) $(\alpha^*) \in \text{RegE}(\Sigma)$ („Repetition“)

Schreibweise 2.2 (Vereinfachung). Präzedenzregeln, um Klammern zu sparen:

- * bindet stärker als $\cdot$
- $\cdot$ bindet stärker als $\vee$

Der Punkt wird weggelassen.

Beispiel 2.3. $a \vee b^* c$ statt $(a \vee ((b^*) \cdot c))$

Definition 2.4 (Semantik von RegE(Σ)**).** Ein regulärer Ausdruck α beschreibt eine formale Sprache $[\![\alpha]\!] = L(\alpha) \subseteq \Sigma^*$:

(i) $L(\Lambda) := \emptyset$

(ii) $L(a) := \{a\}$

(iii) $L(\alpha \vee \beta) := L(\alpha) \vee L(\beta)$

(iv) $L(\alpha \cdot \beta) := L(\alpha) \cdot L(\beta)$

(v) $L(\alpha^*) := L(\alpha)^*$

Sprechweise. $w \in L(\alpha) \curvearrowright w$ ist ein Match für α, α ist ein Muster (Pattern).

Definition 2.5 (Klasse der regulären Sprachen). Die Klasse $\mathrm{RegL}(\Sigma)$ der regulären Sprachen über Σ ist induktiv definiert durch:

- $\emptyset, \{a\} \in \mathrm{RegL}$ für alle $a \in \Sigma$
- $L, L' \in \mathrm{RegL} \curvearrowright L \cup L', LL', L^* \in \mathrm{RegL}(\Sigma)$

Folgerung 2.6. Die Klasse $\mathrm{RegL}(\Sigma) = \mathfrak{L}(\Sigma, \mathrm{RegE})$.

2.2 Deterministische endliche Automaten

Definition 2.7 (deterministischer endlicher Automat). Seien Q und Σ nicht-leere, endliche Mengen, $q_0 \in Q$, $F \subseteq Q$ und $\delta : Q \times \Sigma \to Q$.
Dann heißt $\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$ ein DETERMINISTISCHER ENDLICHER AUTOMAT ÜBER Σ mit der ZUSTANDSMENGE Q, dem EINGABEALPHABET Σ, der TRANSITIONSFUNKTION δ, dem ANFANGSZUSTAND q_0 und der ENDZUSTANDSMENGE F.

Bezeichnung 2.8 (DFA). $\mathfrak{A} \in \mathrm{DFA}(\Sigma), \mathfrak{A} \in \mathrm{DFA}$ (det. finite aut.)

Definition 2.9 (erweiterte Transitionsfunktion / erkannte Sprache).
$\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F \rangle \in \mathrm{DFA}(\Sigma)$ bestimmt die ERWEITERTE TRANSITIONSFUNKTION

$$\overline{\delta} : Q \times \Sigma^* \to Q$$

mit

$$\overline{\delta}(q, \varepsilon) := q$$
$$\overline{\delta}(q, wa) := \delta(\overline{\delta}(q, w), a) \text{ für } w \in \Sigma^*, a \in \Sigma$$

und damit die VON $\mathfrak{A}$ ERKANNTE SPRACHE

$$L(\mathfrak{A}) := \left\{ w \in \Sigma^* \mid \overline{\delta}(q_o, w) \in F \right\}$$

Beispiel 2.10 (zu Definition 2.9). $\mathfrak{L}(\Sigma, \mathrm{DFA})$: Klasse der von endlichen Automaten erkennbaren Sprachen über Σ.

Ziel. $\mathfrak{L}(\Sigma, \mathrm{DFA}) = \mathrm{RegL}(\Sigma)$ nachweisen
Hilfsmittel: nicht-deterministische Automaten
Hinweis: Scanner, Suchmaschinen, Software-Tools

2.3 Nicht-deterministische endliche Automaten

Definition 2.11 (nicht-deterministischer endlicher Automat). Seien Q, Σ, q_0 und F wie bei einem $\mathfrak{A} \in$ DFA, ferner $\Sigma_\varepsilon := \Sigma \cup \{\varepsilon\}$ und $\delta : Q \times \Sigma_\varepsilon \to P(Q)$. Dann heißt

$$\mathfrak{A} = \langle Q, \Sigma_\varepsilon, \delta, q_0, F \rangle$$

ein NICHT-DETERMINISTISCHER ENDLICHER AUTOMAT ÜBER Σ.

Bezeichnung 2.12 (NFA). NFA(Σ), NFA. Es folgt: DFA$(\Sigma) \subseteq$ NFA(Σ).

Semantik (Erweiterung der Semantik von DFA). Für $T \subseteq Q$ ist die ε-HÜLLE VON T, Bezeichnung $\varepsilon(T)$, induktiv definiert durch:

(i) $T \subseteq \varepsilon(T)$

(ii) $q \in \varepsilon(T) \curvearrowright \delta(q, \varepsilon) \subseteq \varepsilon(T)$

Die ERWEITERTE TRANSITIONSFUNKTION

$$\overline{\delta} : P(Q) \times \Sigma^* \to P(Q)$$

ist induktiv definiert durch:

$$\begin{aligned} \overline{\delta}(T, \varepsilon) &:= \varepsilon(T) \\ \overline{\delta}(T, wa) &:= \varepsilon\Big(\bigcup_{q \in \overline{\delta}(T,w)} \delta(q, a) \Big) \end{aligned}$$

Damit ist die DURCH $\mathfrak{A}$ ERKANNTE SPRACHE definiert als:

$$L(\mathfrak{A}) := \left\{ w \in \Sigma^* | \overline{\delta}(\{q_0\}, w) \cap F \neq \emptyset \right\}$$

Beachte. Für DFAs stimmen beide Semantik-Definitionen überein:

$$\begin{aligned} &| \delta(q, a) | = 1 \text{ für alle } q \in Q, a \in \Sigma \\ \text{und } &| \delta(q, \varepsilon) | = 0 \end{aligned}$$

Definition 2.13 (Potenzmengenautomat). Sei $\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F \rangle \in$ NFA(Σ). Der POTENZMENGENAUTOMAT $\mathfrak{A}^P := \left\langle \hat{Q}, \Sigma, \hat{\delta}, \hat{q}_0, \hat{F} \right\rangle \in$ DFA(Σ) ist definiert durch:

- $\hat{Q} := \left\{ T \subseteq Q | T = \hat{\delta}(\{q_0\}, w), w \in \Sigma^* \right\}$
- $\hat{\delta} := \hat{Q} \times \Sigma \to \hat{Q}$ mit $\hat{\delta}(T, a) := \overline{\delta}(T, a)$
- $\hat{q}_0 := \varepsilon(\{q_0\})$

- $T \in \hat{F} \curvearrowright T \in \hat{Q}$ und $T \cap F \neq \emptyset$

Beweis. $T \in \hat{Q} \curvearrowright T = \overline{\delta}(\{q_0\}, w)$
Ferner: $\overline{\delta}(T, \varepsilon) = \varepsilon(T) = T$
$\overline{\delta}(T,a) = \varepsilon(\bigcup_{q \in \overline{\delta}(T,\varepsilon)} \delta(q,a)) = \varepsilon(\bigcup_{q \in T} \delta(q,a)) = \varepsilon(\bigcup_{q \in \overline{\delta}(\{q_0\},w)} \delta(q,a)) = \overline{\delta}(\{q_0\}, wa) \in \hat{Q}$ □

Lemma 2.14. Für $\mathfrak{A} \in \mathrm{NFA}(\Sigma)$ gilt:

$$L(\mathfrak{A}) = L(\mathfrak{A}^P)$$

Beweis.

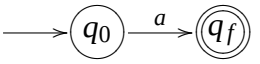
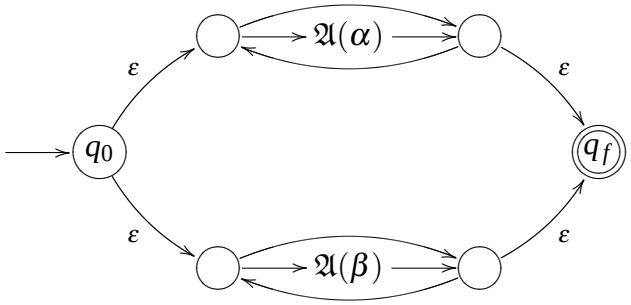
$$\begin{aligned} w \in L(\mathfrak{A}) &\curvearrowright \overline{\delta}(\{q_0\}, w) \cap F \neq \emptyset \\ &\curvearrowright \overline{\delta}(\varepsilon(\{q_0\}), w) \cap F \neq \emptyset \\ &\curvearrowright \hat{\overline{\delta}}(\hat{q}_0, w) \in \hat{F} \\ &\curvearrowright w \in L(\mathfrak{A}^P) \end{aligned}$$ □

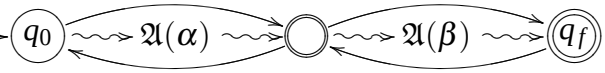
2.4 Synthese und Analyse endlicher Automaten

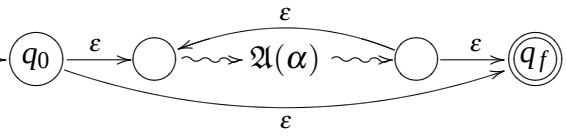
2.4.1 Synthese endlicher Automaten

SYNTHESE:
Konstruiere für $\alpha \in \mathrm{RegE}(\Sigma)$ einen äquivalenten $\mathfrak{A}(\alpha) \in \mathrm{NFA}(\Sigma)$, d.h. $L(\alpha) = L(\mathfrak{A}(\alpha))$.

Algorithmus 2.15 (Thompson). Idee: $\mathfrak{A}(\alpha)$ hat genau einen Endzustand $q_f \neq q_0$. q_0 ist Quelle, q_f ist Senke (im Zustandsgraphen).

- $\mathfrak{A}(\Lambda) :=$  (keine Transitionen)
- $\mathfrak{A}(a) :=$  $(a \in \Sigma)$
- $\mathfrak{A}(\alpha \vee \beta) :=$ 

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

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

Offensichtlich gilt für jedes $\alpha \in \mathrm{RegE}$:

$$L(\alpha) = L(\mathfrak{A}(\alpha))$$

Korollar 2.16. $\mathrm{RegL}(\Sigma) \subseteq \mathfrak{L}(\Sigma, \mathrm{DFA})$

2.4.2 Analyse endlicher Automaten

ANALYSE:
Konstruiere für $\mathfrak{A} \in \mathrm{DFA}(\Sigma)$ einen $\alpha(\mathfrak{A}) \in \mathrm{RegE}(\Sigma)$ mit $L(\alpha(\mathfrak{A})) = L(\mathfrak{A})$, oder kurz: $\alpha(\mathfrak{A}) \sim \mathfrak{A}$.
$\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F\rangle \in \mathrm{DFA}(\Sigma)$ und $Q = \{q_1, \ldots, q_n\}$
Für $i, j \in \{1, \ldots, n\}$ und $k \in \{0, \ldots, n\}$ definieren wir:

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

Dann gilt:

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

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

(1) $k = 0 : W_{ij}^0 \subseteq \Sigma \cup \{\varepsilon\}$

$\curvearrowright W_{ij}^0$ regulär; beachte: $L(\Lambda^*) = \{\varepsilon\}$

(2) $k-1 \to k$:

$W_{ij}^k = W_{ij}^{k-1} \cup W_{ik}^{k-1}(W_{kk}^{k-1})^* W_{kj}^{k-1}$ für $k \in \{1, \ldots, n\}$

Korollar 2.17 (Satz von Kleene).

$$\mathfrak{L}(\Sigma, \mathrm{DFA}) = \mathfrak{L}(\Sigma, \mathrm{RegE}) = \mathrm{RegL}(\Sigma) = \text{Typ 3-Sprachen (Chomsky)}$$

2.4.3 Wortsuche in Texten

Problem: Bestimme alle Dokumente, in denen ein Wort einer gegebenen Wortmenge vorkommt.

Beispiel 2.18. Der Suchautomat von $\{\text{web, ebay}\}$. Es gibt zwei Implementierungstechniken:

(1) NFA-Methode:

Berechne für ein $n \in \Sigma^*_{\text{ASCII}}$ einen Lauf durch den Suchautomaten (erreichbare Zustandsmengen).

Beispiel: $n =$ aebwewebba

$\{1\}$ a $\{1\}$ e $\{1,5\}$ b $\{1,6\}$ w $\{1,2\}$ e $\{1,3,5\}$ w $\{1,2\}$ e $\{1,3,5\}$ b $\{1,\boxed{\boxed{4}}\}$ $\rightarrow$ erkannt!

(2) DFA-Methode:

Konstruiere den Potenzmengen-Automaten des Suchautomaten. □

Bemerkung 2.19. Die Implementierungen von `egrep` und `fgrep` verwenden beide Techniken.

2.5 Pumping-Lemma

Das Pumping-Lemma ist ein Hilfsmittel zum Nachweis nicht-regulärer Sprachen.

Satz 2.20 (Pumping-Lemma, Iterationslemma). Sei $L \in \text{RegL}(\Sigma)$. Dann existiert $k \in \mathbb{N}$, so dass für jedes $x \in L$ mit $|\, x \,| \geq k$ eine Zerlegung

$$x = uvw$$

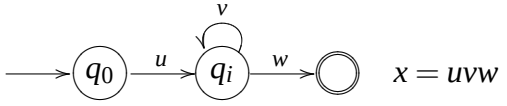
mit folgenden Eigenschaften existiert:

(i) $|\, v \,| \geq 1$

(ii) $|\, uv \,| \leq k$

(iii) $uv^i w \in L$ für alle $i \in \mathbb{N}$ (auch $i = 0$)

Beweis. Sei $\mathfrak{A} \in \text{DFA}(\Sigma)$ mit $L(\mathfrak{A}) = L$ und $k :=| Q |$.
Sei $x \in L$ mit $|\, x \,| \geq k$. Dann wird beim „Erkennungslauf“ von x in $\mathfrak{A}$ mindestens ein Zustand mehr als einmal besucht:



$x = uvw$

Sei q_i der erste solche Iterationszustand und v Teilwort von dort bis zur ersten Wiederholung. Dann gilt:

$$| uv | -1 \leq k-1 \curvearrowright \text{(ii)}$$

(i) und (iii) klar. □

Beachte. Das Pumping-Lemma beschreibt eine notwendige, aber nicht eine hinreichende Eigenschaft.

Beispiel 2.21.

$$L = \{a^n b^n | n \geq 1\} \notin \text{RegL}(\{a,b\})$$

Beweis. Angenommen, $L \in$ RegL.
Dann existiert $k \in \mathbb{N}$ mit (i)-(iii) aus Pumping-Lemma. Für $x = a^k b^k$ muss eine Zerlegung existieren:

$$a^k b^k = uvw$$

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!
Also: $L \notin$ RegL. □

2.6 Zustandsreduktion endlicher Automaten

Ziel: Konstruktion endlicher Automaten mit minimaler Zustandszahl.

Definition 2.22 (Ableitung). Für $L \subseteq \Sigma^*$ und $w \in \Sigma^*$ heißt

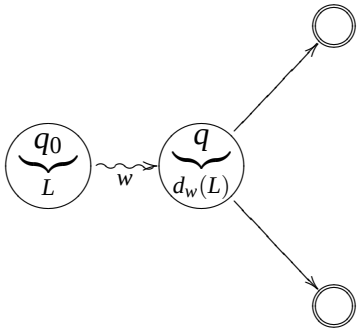
$$d_w(L) := \{v \in \Sigma^* | wv \in L\}$$

die ABLEITUNG von L nach w.

Lemma 2.23. Sei $L = L(\mathfrak{A})$ für $\mathfrak{A} = \langle Q, \Sigma, \delta, q_o, F \rangle \in \text{DFA}(\Sigma)$. Dann gilt:

$$D(L) = \{d_w(L) | w \in \Sigma^*\} : | D(L) | \leq | Q |$$

Beweis.



Für $q \in Q$ bezeichne:

$$L(q) := L(\langle Q, \Sigma, \delta, q, F \rangle)$$

Dann gilt:

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

Definition 2.24 (Ableitungsautomat). Sei $L \in \text{RegL}$.
Der ABLEITUNGSAUTOMAT $\mathfrak{A}_L = \langle D(L), \Sigma, \delta, q_0, F \rangle \in \text{DFA}(\Sigma)$ ist definiert durch:

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

Beachte.

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

Also ist die Definition 2.24 unabhängig vom Repräsentanten w.

Lemma 2.25.

$$L(\mathfrak{A}_L) = L \quad \text{(d.h. } \mathfrak{A}_L \text{ erkennt } L\text{)}$$

Beweis.

$$\begin{aligned} w \in L(\mathfrak{A}_L) &\curvearrowright \overline{\delta}(q_0, w) \in F \\ &\curvearrowright d_w \in F \\ &\curvearrowright w \in L \end{aligned}$$ □

Korollar 2.26. (i) Der Ableitungsautomat ist zustandsminimal.

(ii) Für $L \subseteq \Sigma^*$ gilt:

$$L \in \mathrm{RegL} \curvearrowright D(L) < \infty$$

ZUSTANDSREDUKTION: Konstruktion eines zustandsminimalen Automaten aus einem gegebenen Automaten durch:

- Weglassen nicht erreichbarer Zustände
- Verschmelzen äquivalenter Zustände

Dabei heißt:

- $q \in Q$ ERREICHBAR: $\curvearrowright \exists w \in \Sigma^* : \overline{\delta}(q_0, w) = q$
- $q_1 \sim q_2$ (ÄQUIVALENT) $\curvearrowright L(q_1) = L(q_2)$

Definition 2.27 (Faktorautomat). Für $\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F \rangle \in \mathrm{DFA}(\Sigma)$ ist der FAKTORAUTOMAT

$$\mathfrak{A}/_\sim := \left\langle \tilde{Q}, \Sigma, \tilde{\delta}, \tilde{q_0}, \tilde{F} \right\rangle \in \mathrm{DFA}(\Sigma)$$

wie folgt definiert: (Für $q \in Q$ sei $[q] := \{q' | q' \sim q\}$.)

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

Erinnerung (Zustandsminimierung von endlichen Automaten).
- $L \in \mathrm{RegL}(\Sigma) \mapsto \mathfrak{A}_L$: Ableitungsautomat
- $\mathfrak{A}/_\sim$: Faktorautomat

Beachte. $q \in F, q \sim q' \curvearrowright q' \in F$ und $q \sim q' \curvearrowright \delta(q, a) \sim \delta(q', a)$ zeigt die Unabhängigkeit der Definition 2.27 von den Repräsentanten.

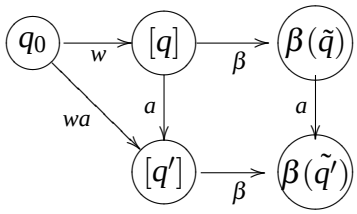
Lemma 2.29. Für $\mathfrak{A} \in \mathrm{DFA}(\Sigma)$ gilt:

$$\mathfrak{A}/_\sim = \mathfrak{A}_{L(\mathfrak{A})} \quad \text{(bis auf Zustandsnamen)}$$

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

Beweis. Sei $\beta : \tilde{Q} \to D(L(\mathfrak{A}))$ definiert durch:

$$\beta([\overline{\delta}(q_0, w)]) := d_w(L(\mathfrak{A}))$$ □

- β ist unabhängig vom Repräsentanten $w \in \Sigma^*$:
  $\overline{\delta}(q_0, v) \sim \overline{\delta}(q_0, w) \curvearrowright L(\overline{\delta}(q_0, w)) = L(\overline{\delta}(q_0, v)) \curvearrowright d_w(L(\mathfrak{A})) = d_v(L(\mathfrak{A}))$ $(*)$
- β ist bijektiv, weil die Folgerung $(*)$ umkehrbar ist
- β ist strukturerhaltend:
  - $\beta([q_0]) = \beta([\overline{\delta}(q_0, \varepsilon)]) = d_\varepsilon(L(\mathfrak{A})) = L(\mathfrak{A})$ (Anfangszustand des Ableitungsautomaten)
  - $\overline{\delta}(q_0, w) = q \in F$:
    $\beta([\overline{\delta}(q_0, w)]) = d_w(L(\mathfrak{A}))$ ist ebenfalls Endzustand im Ableitungsautomaten
  - 

$\overline{\delta}(\underbrace{[q]}_{=[\overline{\delta}(q_0,w)]}, a) = \underbrace{[\delta(q,a)]}_{=[\overline{\delta}(q_0,wa)]}$ $\qquad =\delta(d_w(L(\mathfrak{A})),a) \qquad =d_{wa}(L(\mathfrak{A}))$ □

Verfahren zur Zustandsminimierung:

- Weglassen nicht erreichbarer Zustände
- Verschmelzen äquivalenter Zustände

Definition 2.30 (k-äquivalent). Sei $\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F \rangle \in$ DFA und jeder Zustand erreichbar; ferner sei $k \in \mathbb{N}, q_1, q2 \in Q$. Dann sagt man:

- q_1 k-ÄQUIVALENT q_2 $(q_1 \overset{k}{\sim} q_2) \curvearrowleft \forall w \in \Sigma^*, \mid w \mid \leq k : w \in L(q_1) \curvearrowleft w \in L(q_2)$
- $\mathfrak{A}$ INDUZIERT die Abbildungen $r^k : Q^2 \to \{0,1\}$ mit $r^k(q_1, q_2) = 1 :\curvearrowleft q_1 \overset{k}{\sim} q_2$.
  Sie können als Matrizen $R^k = (r^k(q_i, q_j))_{i,j=1}^n$ mit $n = \mid Q \mid$ dargestellt werden.

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

Berechnung der k-Äquivalenz-Matrizen:

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

Lemma 2.31. 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 $r^k = r^{k+n}$ gelten für alle $n \in \mathbb{N}$.
Zu zeigen: $r^{k+1} = r^{k+2}$.
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 $a_1 \dots a_{k+2} \in L(q_1)$.
Dann gilt: $\delta(q_1,a_1) \overset{k}{\sim} \delta(q_2,a_1)$. Und nach $(*)$ gilt auch: $\delta(q_1,a_1) \overset{k+1}{\sim} \delta(q_2,a_1)$.
Somit gilt: $a_2 \dots a_{k+2} \in L(\delta(q_2,a_1))$ und $a_1 \dots a_{k+2} \in L(q_2)$. □

Algorithmus 2.32 (Markierungsalgorithmus).
Eingabe: $\mathfrak{A} = \langle Q,\Sigma,\delta,q_o,F\rangle \in \mathrm{DFA}(\Sigma)$, jedes $q \in Q$ ist erreichbar.
Ausgabe: äquivalente Zustände
Verfahren:

- Stelle eine Tabelle aller Zustandspaare $\{q,q'\}$ mit $q \neq q'$ auf.
- Markiere alle Paare $\{q,q'\}$ 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.
- Die unmarkierten Paare repräsentieren äquivalente Zustände.

Bemerkung 2.33. Die Implementierung des Markierungsalgorithmus ist mit $O(\mid q \mid^2)$ Zeitkomplexität möglich.

2.7 Entscheidbare Eigenschaften

(a) Deterministische endliche Automaten:

- Wortproblem: $w \in L(\mathfrak{A})$? ($w \in \Sigma^*$ und $\mathfrak{A} \in \mathrm{DFA}(\Sigma)$)
  Durch Eingabe und Zustandsüberprüfung in $\mid w \mid + 1$ Schritten entscheidbar.
- $\emptyset$-Problem: $L(\mathfrak{A}) = \emptyset$? ($\mathfrak{A} \in \mathrm{DFA}(\Sigma)$)
  Durch Eingabe aller Wörter w mit $\mid w \mid < \mid Q \mid$ entscheidbar.
  Beachte: $w \in L(\mathfrak{A}), \mid w \mid \geq \mid Q \mid \curvearrowright \exists v \in L(\mathfrak{A}), \mid v \mid < \mid w \mid$.
  Verfahren: Testen, ob F von q_0 erreichbar.
  Durchführung in $O(\mid q \mid^2)$-Zeit

- $\sim$-Problem: $L(\mathfrak{A}) = L(\mathfrak{A}')$? ($\mathfrak{A}, \mathfrak{A}' \in \mathrm{DFA}(\Sigma)$)
  Reduktion auf $\emptyset$-Problem:

$$L(\mathfrak{A}) = L(\mathfrak{A}') \curvearrowleft L(\mathfrak{A}) \subseteq L(\mathfrak{A}') \text{ und } L(\mathfrak{A}') \subseteq L(\mathfrak{A})$$
$$\curvearrowleft L(\mathfrak{A}) \cap \overline{L(\mathfrak{A}')} = \emptyset \text{ und } L(\mathfrak{A}') \cap \overline{L(\mathfrak{A})} = \emptyset$$

  2 Produkt-Automaten (vgl. Übung 4) mit $| Q | \cdot | Q' |$ Zuständen, testen auf $\emptyset$
  $\sim$-Problem in $O(| Q |^2 \cdot | Q' |^2)$-Zeit entscheidbar

(b) Reguläre Ausdrücke, nicht-deterministische endliche Automaten:

Durch Transformation in DFAs sind alle drei Probleme entscheidbar, aber der Aufwand steigt.

- Wortproblem: $O(| \alpha | \cdot | w |)$- bzw. $O(| Q |^4 \cdot | w |)$-Zeit
- $\emptyset$-Problem: NFA wie DFA behandeln
  RegE $\mapsto$ NFA in $O(| \alpha |)$-Zeit mit $| Q | \leq 2 \cdot | \alpha |$ entscheidbar
- $\sim$-Problem: NP-hart (exponentieller Aufwand, Potenzmengenkonstruktion) □

2.8 Endliche Automaten mit Ausgabe

Idee: Erweiterung eines DFA um ein Ausgabeband
Ziel: Nicht Wörter erkennen, sondern Wörter transformieren, und Berechnung von Wortfunktionen.

Definition 2.34 (allgemein-sequentieller Automat). Sei $\langle Q, \Sigma, \delta, q_0, F \rangle \in \mathrm{DFA}$, Δ ein AUSGABEALPHABET und $\lambda : Q \times \Sigma \to \Delta^*$ eine AUSGABEFUNKTION.
Dann heißt $\mathfrak{A} = \langle Q, \Sigma, \Delta, q_0, \delta, \lambda \rangle$ ein ALLGEMEIN-SEQUENTIELLER AUTOMAT.

Bezeichnung 2.35 (für allgemein-sequentiellen Automaten). $\mathfrak{A} \in \mathrm{GSM}(\Sigma, \Delta)$
(„generalized sequential machine“)

Beachte. Beim allgemein-sequentiellen Automaten werden keine Endzustände berücksichtigt.

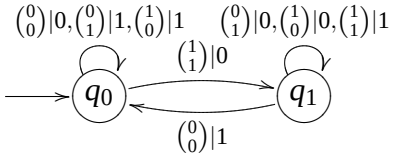
Graphische Darstellung: $(q) \xrightarrow{a|w} (q')$ falls $\delta(q,a) = q'$ und $\lambda(q,a) = w$
$\mathfrak{A}$ berechnet eine Wertefunktion $f_{\mathfrak{A}} : \Sigma^* \to \Delta^*$.
Wir erweitern λ zu $\overline{\lambda} : Q \times \Sigma^* \to \Delta^*$ durch $\overline{\lambda}(q, \varepsilon) = \varepsilon$, $\overline{\lambda}(q, wa) := \overline{\lambda}(q, w) \cdot \lambda(\overline{\delta}(q, w), a)$.
Dann ist $f_{\mathfrak{A}} := \overline{\lambda}(q_o, w)$.

Definition 2.36 (allgemein-sequentielle Funktion). $f : \Sigma^* \to \Delta^*$ heißt ALLGEMEIN-SEQUENTIELL, wenn es $\mathfrak{A} \in \mathrm{GSM}(\Sigma, \Delta)$ gibt mit $f = f_{\mathfrak{A}}$.
Spezialfall: Sei $\mathfrak{A} \in \mathrm{GSM}(\Sigma, \Delta)$. Falls $\lambda : Q \times \Sigma \to \Delta$, so heißt $\mathfrak{A}$ auch MEALY-AUTOMAT und $f_{\mathfrak{A}}$ SEQUENTIELL.

Was können diese Automaten?

Beispiel 2.37 (binäre Addition). Addition gespiegelter Binärzahlen (mit wenigstens einer Führungsnull):
$\Sigma = \mathbb{B}^2, \Delta = \mathbb{B}$ mit $\mathbb{B} = \{0,1\}$

	0	1	1	0	1	0	1
+	0	1	0	1	1	0	1
=	1	1	0	0	0	1	0



Satz 2.38 (Ginsberg / Rose). Sei $f : \Sigma^* \to \Delta^*$. Dann gilt:

f ist allgemein-sequentiell $\curvearrowleft$ f erfüllt (i) bis (iv)

(i) $f(\varepsilon) = \varepsilon$

(ii) f ist PRÄFIXTREU, d.h.

$$\forall u, v \in \Sigma^* \; \exists w \in \Delta^* \text{ mit } f(uv) = f(u)w$$

(iii) f ist LÄNGENBESCHRÄNKT, d.h.

$$\exists k \in \mathbb{N} \text{ mit } \mid f(w) \mid \leq k \mid w \mid \;\; \forall w \in \Sigma^*$$

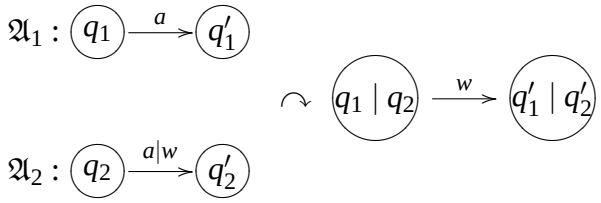
(iv) f erhält REGULARITÄT, d.h.

(a) $L \in \mathrm{RegL}(\Sigma) \curvearrowright f(L) := \{f(w) | w \in L\} \in \mathrm{RegL}(\Delta)$

(b) $L \in \mathrm{RegL}(\Delta) \curvearrowright f^{-1}(L) := \{w \in \Sigma^* | f(w) \in L\} \in \mathrm{RegL}(\Sigma)$

Beweis. „$\curvearrowright$“: (i) bis (iii) folgen aus der Definition von $f_{\mathfrak{A}} = f$.
(iv)(a): $L \in \mathrm{RegL}(\Sigma) : \mathfrak{A}_1 = \langle Q_1, \Sigma, \delta_1, q_0^1, F_1 \rangle \in \mathrm{DFA}(\Sigma)$ mit $L = L(\mathfrak{A})$.
f allgemein-sequentiell: $\mathfrak{A}_2 = \langle Q_2, \Sigma, \Delta, q_0^2, \delta_2, \lambda_2 \rangle$
Zu zeigen: $f(L) \in \mathrm{RegL}(\Delta)$.
Konstruktion von $\mathfrak{A} \in \mathrm{NFA}(\Delta)$ mit $L(\mathfrak{A}) = f(L)$ durch Parallelkonstruktion und Selektion der Ausgabe.
Wortübergänge ($q \xrightarrow{a} \circ \xrightarrow{b} \circ \xrightarrow{c} q'$) können zugelassen werden (Zwischenzustände ausführen):
$\mathfrak{A} = \langle Q, \Delta, \delta, q_0, F \rangle$ mit $\delta : Q \times \Delta^* \dashrightarrow P(Q)$ mit $\mid \mathrm{Def}(\delta) \mid < \infty$, $Q := Q_1 \times Q_2$, $q_0 := (q_0^1, q_0^2)$, $F := F_1 \times Q_2$.

$\delta((q_1,q_2),w) \ni (q_1',q_2') :\curvearrowleft$ es ex. $a \in \Sigma$ mit $\delta_1(q_1,a) = q_1'$ und $\delta_2(q_2,a) = q_2'$ und $\lambda_2(q_2,a) = w$



Es folgt: $w \in L(\mathfrak{A}) \curvearrowleft$ es ex. $v \in L$ mit $f(v) = w$. □

Folgerung 2.39. (1) Die Spiegelfunktion $\cdot^R : \Sigma^* \to \Sigma^*$ ist nicht allgemein-sequentiell.

(2) Die serielle Multiplikation von Binärzahlen ist nicht allgemein-sequentiell.

Beweis. (1) $\cdot^R$ erfüllt nicht die Präfixtreue.

(2) Die serielle Multiplikation von Binärzahlen sei definiert durch

$$f : (\mathbb{B}^2)^* \to \mathbb{B}^*$$

mit

$$f((a_1,b_1)(a_2,b_2)\cdots(a_n,b_n)) = a_1a_2\cdots a_n \underbrace{\cdot}_{\text{binär}} b_1b_2\cdots b_n$$

(durch Führungsnullen gleiche Länge).

f erfüllt nicht (iv)(a), d.h. erhält nicht Regularität.

Sei $L \subseteq (\mathbb{B}^2)^*$ gegeben durch $(1,0)(0,1)^+ \in \mathrm{RegE}(\mathbb{B}^2)$. $w = (1,0)(0,1)^k \in L$ stellt das Binärzahlenpaar $(10^k, 01^k)$ dar, welches die Zahlen 2^k und $2^k - 1$ repräsentiert, da $2^k \cdot (2^k - 1) = 2^{2k} - 2^k$ die Binärdarstellung 1^k0^k hat.

Also gilt: $f(L) = \{1^k0^k | k \geq 1\} \notin \mathrm{RegL}(\mathbb{B})$. □

3 Kontextfreie Sprachen und Kellerautomaten

3.1 Kontextfreie Grammatiken

Definition 3.1 (kontextfreie Grammatik). Seien N und Σ nicht-leere endliche Mengen mit $N \cap \Sigma = \emptyset$.
Sei $P \subseteq N \times (N \cup \Sigma)^*$ mit $\mid P \mid < \infty$ und $S \in N$.
Dann heißt $G = \langle N, \Sigma, P, S \rangle$ eine KONTEXTFREIE GRAMMATIK.

Bezeichnung 3.2 (kontextfreie Grammatik). $G \in \mathrm{CFG}(\Sigma)$ oder $G \in \mathrm{CFG}$

Bezeichnungen und Konventionen.

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

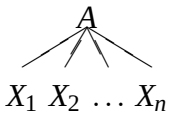
Definition 3.3 (Ableitungsrelation und Ableitungsschritt). Sei $G = \langle N, \Sigma, P, S \rangle \in \mathrm{CFG}$ und $\pi = A \to \alpha \in P$.
π bestimmt eine ABLEITUNGSRELATION $\Rightarrow_\pi \subseteq \mathcal{X}^* \times \mathcal{X}^*$ mit $\beta_1 \Rightarrow_\pi \beta_2 :\curvearrowright$ es ex. 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.
G bestimmt die Ableitungsrelation $\Rightarrow_G \subseteq \mathcal{X}^* \times \mathcal{X}^*$ durch $\Rightarrow_G := \bigcup_{\pi \in P} \Rightarrow_\pi$ und damit DIE VON G ERZEUGTE SPRACHE $L(G) := \{w \in \Sigma^* | S \overset{*}{\Rightarrow}_G w\}$.
$\overset{*}{\Rightarrow}_G := \bigcup_{n=0}^{\infty} \overset{n}{\Rightarrow}_G$ (reflexive und transitive Hülle von $\Rightarrow_G$)
Es folgt: $w \in L(G) \curvearrowright \exists \alpha_0, \alpha_1, \ldots, \alpha_n \in \mathcal{X}^*$ und $\pi_1, \ldots, \pi_n \in P$: $S = \alpha_0 \Rightarrow_{\pi_1} \alpha_1 \Rightarrow_{\pi_2} \alpha_2 \ldots \Rightarrow_{\pi_n} \alpha_n = w$.

Bezeichnung 3.4. $\mathrm{CFL}(\Sigma) := \{L \subseteq \Sigma^* | \exists G \in \mathrm{CFG}(\Sigma) : L(G) = L\}$

3.1.1 Ableitungsbäume, Rechts- und Linksableitungen, Ein- und Mehrdeutigkeit

Sei $G = \langle N, \Sigma, P, S \rangle \in \mathrm{CFG}$.
Darstellung von Ableitungen durch Bäume:

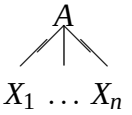
(1) Eine Regel $\pi = A \to X_1 \dots X_n$ bestimmt den REGELBAUM:



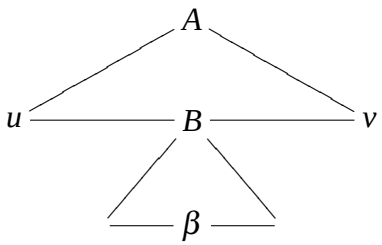
Spezialform: $A \to \varepsilon$

(2) Eine Ableitung $A \Rightarrow \alpha_1 \dots \Rightarrow \alpha_n$ bestimmt den ABLEITUNGSBAUM durch entsprechendes Verkleben der Regelbäume. Er lässt sich durch Induktion über n definieren:

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



$\alpha_n \Rightarrow \alpha_{n+1}$ sei gegeben durch den Ableitungsschritt (u, π, v). Dann entsteht der Ableitungsbaum von $A \Rightarrow \alpha_1 \Rightarrow \dots \Rightarrow \alpha_n \Rightarrow \alpha_{n+1}$ durch Anhängen des Regelbaums von π zwischen u und v:



Regel $\pi = B \to \beta$

Folgerung 3.5. Ein Ableitungsbaum repräsentiert eine Klasse von Ableitungen, die sich nur in der Reihenfolge von Ableitungsschritten unterscheiden. Ein Ableitungsbaum repräsentiert die relevante syntaktische Struktur.

Definition 3.6 (Rechtsableitung/Linksableitung). Eine Ableitung heißt RECHTSABLEITUNG (bzw. LINKSABLEITUNG), wenn jeder Ableitungsschritt (α, π, β) ein RECHTSABLEITUNGSSCHRITT ist, d.h. $\beta \in \Sigma^*$ (bzw. wenn jeder Ableitungsschritt (α, π, β) ein LINKSABLEITUNGSSCHRITT ist, d.h. $\alpha \in \Sigma^*$).

Schreibweise 3.7. $\underset{l}{\Rightarrow}$ bzw. $\underset{r}{\Rightarrow}$ für Links- bzw. Rechtsableitungsschritte.

Folgerung 3.8. Ein Ableitungsbaum hat genau eine Rechts- und genau eine Linksableitung.

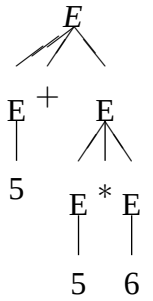
Definition 3.9 (ein-/mehrdeutige kontextfreie Grammatik).

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

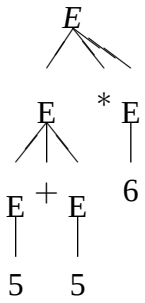
- G heißt MEHRDEUTIG: $\not\circlearrowright G$ ist nicht eindeutig

Beispiel 3.10. $E \to E+E \mid E*E \mid 5 \mid 6$ ist mehrdeutig. Z.B. lässt sich $5+5*6$ mit den beiden folgenden Rechtsableitungen darstellen:

(1) $E \underset{r}{\Rightarrow} E+E \underset{r}{\Rightarrow} E+E*E \overset{3}{\underset{r}{\Rightarrow}} 5+5*6$



(2) $E \underset{r}{\Rightarrow} E*E \underset{r}{\Rightarrow} E*6 \underset{r}{\Rightarrow} E+E*6 \underset{r}{\Rightarrow} E+5*6 \underset{r}{\Rightarrow} 5+5*6$



$E \to (E+E) \mid (E*E) \mid 5 \mid 6$ ist eindeutig. (2) entspricht folgender Linksableitung: $E \underset{l}{\Rightarrow} E*E \underset{l}{\Rightarrow} E+E*E \overset{3}{\underset{l}{\Rightarrow}} 5+5*6$.

3.2 Einseitig-lineare Grammatiken

Definition 3.11 (links-/rechts-/einseitig-linear). Sei $G = \langle N, \Sigma, P, S \rangle \in \text{CFG}$.

- Dann heißt G LINKS-LINEAR, wenn für jedes $\pi = A \to \alpha \in P$ gilt: $\alpha = Bw$ oder $\alpha = w$ für ein $B \in N$ und ein $w \in \Sigma^*$.
- Gilt stattdessen $\alpha = wB$ oder $\alpha = w$ für jedes π, so heißt G RECHTS-LINEAR.
- G heißt EINSEITIG-LINEAR: $\not\circlearrowright G$ ist rechts-linear oder G ist links-linear

Ziel. $\{L(G) | G \text{ links-linear}\} = \{L(G) | G \text{ rechts-linear}\} = \{L(G) | G \text{ einseitig-linear}\} = \text{RegL}(\Sigma)$

Satz 3.12. $\mathfrak{L}(\Sigma, \text{NFA}) = \{L \subseteq \Sigma^* | L = L(G), G \text{ rechts-linear}\}$

Beweis. (1) „$\supseteq$“: Sei $G = \langle N, \Sigma, P, S \rangle$ rechts-linear.

Auffassung von G als $\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F \rangle \in$ NFA:

$Q := N \cup \{q_f\}, q_f$ neu

$q_0 := S$

$F := \{q_f\}$

$\delta(A, w) \ni B$, falls $A \to wB$ in G

$\delta(A, w) \ni q_f$, falls $A \to w$ in G

Offensichtlich: $L(G) = L(\mathfrak{A})$.

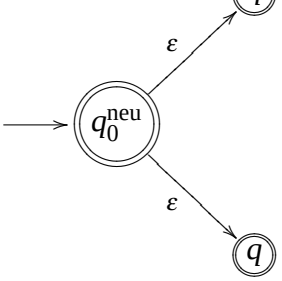
(2) „$\subseteq$“: $\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F \rangle \in$ NFA kann als rechts-lineare Grammatik aufgefasst werden (ähnlich wie oben).

Unterschied: $q \in F \mapsto$ neue Regel $q \to \varepsilon$. □

Korollar 3.13. $\mathrm{RegL}(\Sigma) \subseteq \mathrm{CFL}(\Sigma)$
Für $|\Sigma| \geq 2$ ist diese Inklusion echt, weil $\{a^n b^n | n \in \mathbb{N}\} = L(G)$ mit $G = (S \to aSb | \varepsilon)$.

Lemma 3.14. $L \in \mathrm{RegL} \curvearrowleft L^R \in \mathrm{RegL}$

Beweis. Zu jedem $\mathfrak{A} \in \mathrm{DFA}(\Sigma)$ konstruieren wir $\mathfrak{A}^R \in \mathrm{NFA}(\Sigma)$ mit $L(\mathfrak{A}^R) = L(\mathfrak{A})^R$.
Idee: Ersetze $(q) \xrightarrow{a} (q')$ durch $(q) \xleftarrow{a} (q')$, $\longrightarrow (q_0)$ durch $((q_0))$ (Endzustand) und ergänze diesen NFA durch $((q))$ für alle $q \in F$. □



Korollar 3.15. $\{L \subseteq \Sigma^* | L = L(G), G \text{ rechts-linear}\} = \{L \subseteq \Sigma^* | L = L(G), G \text{ links-linear}\}$

Beweis.

L rechts-linear erzeugbar $\curvearrowleft L \in \mathrm{RegL}$

$\curvearrowleft L^R \in \mathrm{RegL}$

$\curvearrowleft L^R$ rechts-linear erzeugbar

$\curvearrowleft L = L^{RR}$ links-linear erzeugbar (Regeln spiegeln) □

3.3 Normalformen von kontextfreien Grammatiken

Hilfsmittel: Vorgänger von Satzformen $\alpha, \beta \in \mathcal{X}^*$
$\alpha \Rightarrow \beta$: α direkter Vorgänger von β
$\alpha \overset{*}{\Rightarrow} \beta$: α Vorgänger von β

Definition 3.16 (Vorgängermenge/-abschluss). Sei $G = \langle N, \Sigma, P, S \rangle \in \text{CFG}$ und $L \subseteq \mathcal{X}^*$. Dann ist:

(i) die DIREKTE VORGÄNGERMENGE von L definiert durch:
$\underline{\text{Pre}}_{\text{G}}(\text{L}) := \{\alpha \in \mathcal{X}^* | \exists \beta \in L : \alpha \Rightarrow_G \beta\}$

(ii) der VORGÄNGERABSCHLUSS von L definiert durch:
$\underline{\text{Pre}}^*_{\text{G}}(\text{L}) := \{\alpha \in \mathcal{X}^* | \exists \beta \in L : \alpha \overset{*}{\Rightarrow}_G \beta\}$

Lemma 3.17. Sei $G = \langle N, \Sigma, P, S \rangle \in \text{CFG}$ und $L \subseteq \mathcal{X}^*$. Dann gilt:

$$L \in \text{RegL}(\mathcal{X}) \curvearrowright \underline{\text{Pre}}^*_{\text{G}}(\text{L}) \in \text{RegL}(\mathcal{X})$$

Beweis. Sei $\mathfrak{A} = \langle Q, \mathcal{X}, \delta, q_0, F \rangle \in \text{NFA}$ mit $L(\mathfrak{A}) = L$. Konstruiere $\mathfrak{A}' = \langle Q, \mathcal{X}, \delta', q_0, F \rangle \in \text{NFA}$ durch:

(i) $\delta(q,a) \subseteq \delta'(q,a)$ für alle $(q,a) \in Q \times \mathcal{X}_\varepsilon$

(ii) Wenn $A \to \alpha \in P$ und $\overline{\delta'}(\{q\}, \alpha) \ni q'$, so $\delta'(q,A) \ni q'$ hinzufügen.

Ergänze δ' um solche Transitionen, solange dies möglich ist.
Dieser Erweiterungsprozess terminiert, weil Q und $\mathcal{X}$ endlich sind. □

Bemerkung 3.18. Die Zeitkomplexität der Automatenkonstruktion liegt in $\text{O}(\mid Q \mid^6)$.

Definition 3.19 (produktiv/erreichbar). Sei $G \in \text{CFG}(\Sigma)$ und $A \in N$.

(i) A heißt PRODUKTIV: $\curvearrowright \exists w \in \Sigma^* : A \overset{*}{\Rightarrow} w$

(ii) A heißt ERREICHBAR: $\curvearrowright \exists \alpha, \beta \in \mathcal{X}^* : S \overset{*}{\Rightarrow} \alpha A \beta$

Folgerung 3.20. (i) A produktiv $\curvearrowright A \in \underline{\text{Pre}}^*_{\text{G}}(\Sigma^*)$,

(ii) A erreichbar $\curvearrowright S \in \underline{\text{Pre}}^*_{\text{G}}(\mathcal{X}^* A \mathcal{X}^*)$

Definition 3.21 (reduziert). $G \in \text{CFG}(\Sigma)$ heißt REDUZIERT: $\curvearrowright$ entweder $P = \emptyset$ oder jedes $A \in N$ ist produktiv und erreichbar.

Lemma 3.22. Jedes $G \in \mathrm{CFG}(\Sigma)$ lässt sich in eine äquivalente reduzierte Grammatik $G' \in \mathrm{CFG}$ transformieren.

Beweis. Stelle für jedes $A \in N$ fest, ob A produktiv und erreichbar ist (Vorgängerabschluss).

- 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. □

Korollar 3.23. Das $\emptyset$-Problem von CFG's ist entscheidbar.

3.3.1 Elimination von ε-Regeln

Definition 3.24 (ε-frei). $G \in \mathrm{CFG}(\Sigma)$ heißt ε-FREI: $\curvearrowright A \to \varepsilon \in P \curvearrowright A = S$ und S auf keiner rechten Regelseite.

Lemma 3.25. $X \overset{*}{\Rightarrow} \varepsilon \curvearrowright X \in \underline{\mathrm{Pre}}^*(\{\varepsilon\})$

Satz 3.26. Jedes $G \in \mathrm{CFG}(\Sigma)$ lässt sich in eine äquivalente ε-freie Grammatik $G' \in \mathrm{CFG}(\Sigma)$ transformieren.

Beweis. Konstruiere mit Vorgängerabschluss $\underline{\mathrm{Pre}}^*(\{\varepsilon\})$ die Menge $N_\varepsilon := \{A \in N | A \overset{*}{\Rightarrow} \varepsilon\}$ und damit $G' = \langle N', \Sigma, P', S' \rangle \in \mathrm{CFG}$:

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

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

(1) $w = \varepsilon$:

$\varepsilon \in L(G') \curvearrowright S' \to \varepsilon \in P \curvearrowright S \in N_\varepsilon \curvearrowright S \overset{*}{\Rightarrow}_G \varepsilon \curvearrowright \varepsilon \in L(G)$

(2) $w \neq \varepsilon$:

(i) $w \in L(G') \curvearrowright S \overset{*}{\Rightarrow}_{G'} w \curvearrowright S \overset{*}{\Rightarrow}_G w \curvearrowright w \in L(G)$, weil Ableitung in G' mit $\alpha_i \overset{*}{\Rightarrow} \varepsilon$ aufgefüllt werden kann.

(ii) $w \in L(G)$:
Wir zeigen durch Induktion über die Ableitungslänge r, dass gilt:

$$A \overset{*}{\Rightarrow}_G w \in \Sigma^+ \curvearrowright A \overset{*}{\Rightarrow}_{G'} w$$

$r = 1$: $A \Rightarrow_G w, w \neq \varepsilon \curvearrowright A \to w \in P \curvearrowright A \to w \in P' \curvearrowright A \overset{*}{\Rightarrow}_{G'} w$
$r \mapsto r+1$: $A \Rightarrow X_1 \dots X_k \overset{r}{\Rightarrow}_G w$
Dann existieren Ableitungen $X_i \overset{r_i}{\Rightarrow}_G w_i$ mit $w = w_1 \dots w_k$ und $r_i \leq r$.
Falls $w_i \neq \varepsilon$: $X_i \overset{*}{\Rightarrow}_{G'} w_i$ nach Induktionsvoraussetzung.
Falls $w_i = \varepsilon$: $X_i \in N_\varepsilon$
Durch entsprechendes Löschen entsteht:

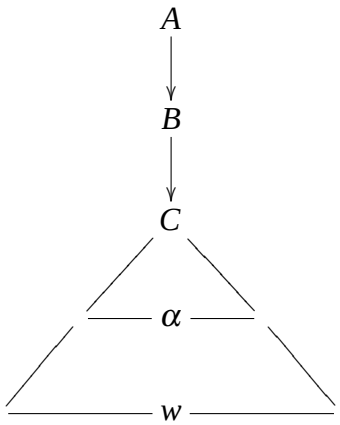
$$A \Rightarrow X'_1 \dots X'_j \overset{*}{\Rightarrow} w'_1 \dots w'_j = w$$ □

3.3.2 Elimination von Kettenregeln

Definition 3.27 (Kettenregel). Eine Regel der Form $A \to B$ heißt KETTENREGEL.

Satz 3.28. Jedes $G = \langle N, \Sigma, P, S \rangle \in$ CFG lässt sich in eine äquivalente ε-freie Grammatik $\overline{G} = \langle \overline{N}, \Sigma, \overline{P}, \overline{S} \rangle \in$ CFG ohne Kettenregeln transformieren.

Beweisidee.



Eliminiere $A \to B$ und $B \to C$. Ergänze $C \to \alpha$ um $B \to \alpha$ und $A \to \alpha$.

Beweis. Transformiere G in äquivalente ε-freie Grammatik $G' = \langle N', \Sigma, P', S' \rangle$.
Dann definieren wir $\overline{G} = \langle \overline{N}, \Sigma, \overline{P}, \overline{S} \rangle$ durch $\overline{N} := N'$, $\overline{S} := S'$, $A \to \alpha \in \overline{P} :\curvearrowright \alpha \notin N'$ und es gibt $B \to \alpha \in P'$ mit $A \in \underline{\text{Pre}}^*(B)$.
Die Korrektheit folgt, da wegen der ε-Freiheit von G' gilt:

$$A \in \underline{\text{Pre}}^*(B) \curvearrowleft A \Rightarrow A_1 \Rightarrow A_2 \dots \Rightarrow A_n = B$$ □

3.3.3 Die Chomsky-Normalform

Definition 3.29 (Chomsky-Normalform). $G = \langle N, \Sigma, P, S \rangle \in \text{CFG}(\Sigma)$ ist in CHOMSKY-NORMALFORM ($G \in \text{CNF}$): $\curvearrowright$ Jede Regel von P hat die Form

- $A \to BC$ mit $B, C \in N$ oder
- $A \to a$ mit $a \in \Sigma$ oder
- $S \to \varepsilon$

Im letzten Fall darf S auf keiner rechten Regelseite auftreten.

Satz 3.30. Jedes $G = \langle N, \Sigma, P, S \rangle \in \text{CFG}$ lässt sich in eine äquivalente Grammatik $G' \in \text{CNF}$ transformieren.

Beweis. O.B.d.A. sei G ε-frei und ohne Kettenregeln.
Außerdem können wir annehmen, dass für $k \geq 2$:

$$A \to X_1 \dots X_k \in P \curvearrowright X_i \in N$$

Denn: $X_i = a$ kann ersetzt werden durch neues $C_a \in N$ unter Hinzufügen von $C_a \to a$.
Bleibt die Elimination von Regeln der Form

$$A \to B_1 \dots B_k \text{ mit } k \geq 3$$

Ersetzen durch:
$A \to B_1 C_1$
$C_1 \to B_2 C_2$
$C_2 \to B_3 C_3$ mit neuen N-Symbolen $C_1, \dots, C_{k-2}$
$\vdots$
$C_{k-2} \to B_{k-1} B_k$ □

3.3.4 Die Greibach-Normalform, Linksrekursion

Definition 3.31 (Greibach-Normalform). $G = \langle N, \Sigma, P, S \rangle \in \text{CFG}(\Sigma)$ ist in GREIBACH-NORMALFORM ($G \in \text{GNF}$): $\curvearrowright$ Jede Regel von P hat die Form

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

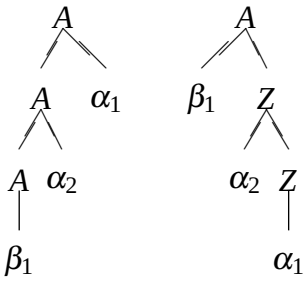
Satz 3.32. Jedes $G \in \text{CFG}$ lässt sich in äquivalente Grammatik $G' \in \text{GNF}$ transformieren.

Beweisidee. Elimination linksrekursiver N-Symbole □

Definition 3.33 (linksrekursiv). $A \in N$ LINKSREKURSIV: $\curvearrowright A \overset{+}{\Rightarrow} A\alpha$ für ein $\alpha \in X^*$

Spezialfall 3.34 (direkte Linksrekursion). Seien $A \to A\alpha_1 \mid A\alpha_2 \mid \ldots \mid A\alpha_r$ (alle Regeln der Form $A \to A\alpha$) und $A \to \beta_1 \mid \ldots \mid \beta_s$.
Dann lässt sich dieser Regelsatz äquivalent ersetzen durch:
$A \to \beta_1 Z \mid \ldots \mid \beta_s Z \mid \beta_1 \mid \ldots \mid \beta_s$
$Z \to \alpha_1 Z \mid \ldots \mid \alpha_r Z \mid \alpha_1 \mid \ldots \mid \alpha_r$, Z neues N-Symbol

Grund für Korrektheit:



Bemerkung 3.35. Linksrekursion stört die Syntaxanalyse.

3.4 Abschlusseigenschaften von CFL

Hilfsmittel: Substitutionssatz, Pumping-Lemma

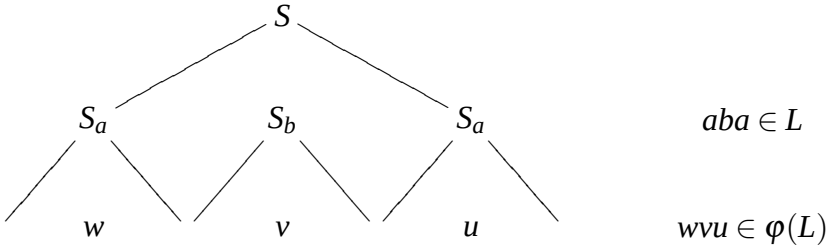
Definition 3.36 (Substitutionsabbildung). Sei Σ ein Alphabet und sei Σ_a für jedes $a \in \Sigma$ ein weiteres Alphabet, $\Delta := \bigcup_{a \in \Sigma} \Sigma_a$.
Dann heißt $\varphi : P(\Sigma^*) \to P(\Delta^*)$ eine SUBSTITUTIONSABBILDUNG, falls

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

Schreibweise 3.37. $\varphi(w) := \varphi(\{w\})$

Satz 3.38 (Substitutionssatz). Sei $L \in \mathrm{CFL}(\Sigma)$, $\varphi(a) \in \mathrm{CFL}(\Sigma_a)$.
Dann ist $\varphi(L) \in \mathrm{CFL}(\Delta)$.

Beweisidee. Kombination der Grammatiken für L und $\varphi(a)$ mit disjunkten N-Symbolen und Ersetzen der $a \in \Sigma$ durch S_a (Startsymbol der Grammatik für $\varphi(a)$). □



$aba \in L$

$wvu \in \varphi(L)$

Korollar 3.39. CFL ist abgeschlossen unter regulären Operationen.

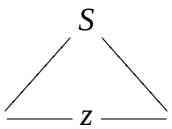
Beweis. Sei $L_1, L_2 \in \mathrm{CFL}(\Sigma)$, $\Sigma' = \{a,b\}$ und $\varphi(a) := L_1$, $\varphi(b) := L_2$. Dann gilt:

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

Da $\{a,b\}, \{ab\}$ und $\{a\}^*$ in $\mathrm{CFL}(\{a,b\})$ liegen, folgt: $L_1 \cup L_2, L_1 L_2, L_1^* \in \mathrm{CFL}(\Sigma)$. □

Der Abschluss unter $\cap$ und $^-$ liegt nicht vor $\rightsquigarrow$ Pumping-Lemma.

Lemma 3.40 (Pumping-Lemma). Sei $L \in \mathrm{CFL}(\Sigma)$.
Dann existiert $k \geq 1$, so dass für alle Worte $z \in L$ mit $\mid z \mid \geq k$ gilt: es existiert eine Zerlegung $z = uvwxy$ mit $\mid vwx \mid \leq k$, $vx \neq \varepsilon$ und $uv^i wx^i y \in L$ für alle $i \in \mathbb{N}$.

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

hat mindestens 2^n Blätter.

Da $G \in \mathrm{CNF}$, muss in t ein Pfad p maximaler Länge mindestens $n+1$ Kanten, also $n+2$ Knoten besitzen. p besitzt daher mindestens $n+1$ Knoten, markiert durch N-Symbole. Also: 2 Knoten mit demselben N-Symbol.
Wähle auf p den letzten Knoten, dessen Marke $A \in N$ sich wiederholt. Dann hat sein Teilbaum höchstens $2^n = k$ Blätter.

- $\mid vwx \mid \leq k$
- $vx \neq \varepsilon$ (Binärstruktur: Verzweigung bei A)
- $uv^i wx^i y$ („Schnippeln“)

□

Lemma 3.41. $L = \{a^n b^n c^n \mid n \geq 1\} \notin \mathrm{CFL}(\{a,b,c\})$

Beweis. Angenommen, L sei kontextfrei. Dann existiert ein Pumping-Index $k \in \mathbb{N}$.
$w = a^k b^k c^k$ besitzt Zerlegung $w = uvwxy$.

$$\left.\begin{array}{l} \mid vwx \mid \leq k \\ vx \neq \varepsilon \\ uwy \in L \end{array}\right\} \begin{array}{l} vx \text{ kann nicht gleichzeitig ein } a \text{ und ein } c \text{ enthalten} \\ \mid uwy \mid < 3k \end{array}$$

Entweder $uwy = a^k \ldots$ oder $uwy = \ldots c^k$.
$\rightsquigarrow$ Widerspruch: $L \in \mathrm{CFL}$. □

Satz 3.42. CFL ist weder unter Durchschnitt noch unter Komplement abgeschlossen.

Beweis. $\left\{ \begin{array}{l} S \to Sc \mid Ac \\ A \to aAb \mid ab \end{array} \right\}$ und $\left\{ \begin{array}{l} S \to aS \mid aA \\ A \to bAc \mid bc \end{array} \right\}$
$L = \{a^n b^n c^p | n, p \geq 1\}, L' = \{a^p b^n c^n | p, n \geq 1\}$
$L, L' \in \text{CFL}, L \cap L' = \{a^n b^n c^n \mid n \geq 1\} \notin \text{CFL}$
Da CFL unter $\cup$ abgeschlossen ist, kann CFL nicht unter $^-$ (Komplement) abgeschlossen sein, denn: $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$. □

3.5 Entscheidbare Eigenschaften von CFG

Satz 3.43. Das Wortproblem und das $\emptyset$-Problem sind für CFG entscheidbar.

Beweis. $w \in L(G) \curvearrowleft S \in \underline{\text{Pre}}^*(\{w\})$
$L(G) = \emptyset \curvearrowleft S \notin \underline{\text{Pre}}^*(\Sigma^*)$ □

Bemerkung 3.44. (i) Das Wortproblem ist in $O(n^3)$ entscheidbar.

(ii) Das Äquivalenzproblem ist nicht entscheidbar.

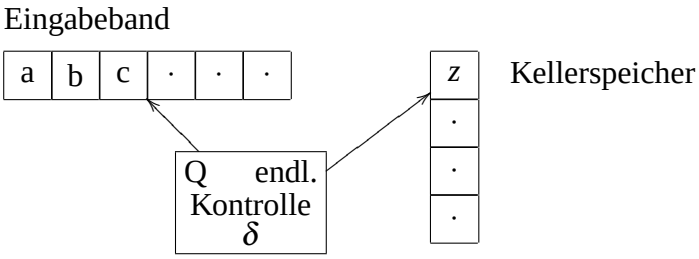
(iii) Die Mehrdeutigkeit ist nicht entscheidbar.

3.6 Kellerautomaten

- KELLERSPEICHER (Stack, Stapel, push down store): wichtige Datenstruktur, „Bindeglied zwischen Wörtern und Bäumen“
- Charakterisierung der 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 / Methoden)

Definition 3.45 (Kellerautomat). Seien Q, Σ und Γ nicht-leere endliche Mengen von ZUSTÄNDEN, EINGABE- und KELLERSYMBOLEN, und seien ferner: $q_0 \in Q$ ANFANGSZUSTAND, $F \subseteq Q$ ENDZUSTANDSMENGE, $Z_0 \in \Gamma$ KELLERSTARTSYMBOL und $\delta : Q \times \Sigma_\varepsilon \times \Gamma \to P_f(Q \times \Gamma^*)$ TRANSITIONSFUNKTION ($\Sigma_\varepsilon := \Sigma \cup \{\varepsilon\}, P_f$: endliche Teilmenge).

Dann heißt $\mathfrak{A} = \langle Q, \Sigma, \Gamma, \delta, q_0, F, Z_0 \rangle$ ein KELLERAUTOMAT ÜBER Σ.



Bezeichnung 3.46 (Kellerautomat). $\mathfrak{A} \in \mathrm{PDA}(\Sigma)$

3.6.1 Semantik

Konfigurationsmenge $Q \times \Sigma^* \times \Gamma^*$ (Kellerspitze links)
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: $(q, w, Z\alpha) \vdash (q', w, \beta\alpha)$, falls $\delta(q, \varepsilon, Z) \ni (q', \beta)$

Beachte. ε-Schritte lassen das Eingabeband unverändert.

3.6.2 Was ist die von $\mathfrak{A}$ erkannte Sprache?

Drei Möglichkeiten der Erkennung:

(a) mit Endzustand

(b) mit leerem Keller

(c) mit beidem

$L(\mathfrak{A}, F) := \{w \in \Sigma^* | (q_0, w, Z_0) \vdash^* (q, \varepsilon, \alpha) \text{ mit } q \in F\}$
$L(\mathfrak{A}, \varepsilon) := \{w \in \Sigma^* | (q_0, w, Z_0) \vdash^* (q, \varepsilon, \varepsilon)\}$
$L(\mathfrak{A}, F, \varepsilon) := \{w \in \Sigma^* | (q_0, w, Z_0) \vdash^* (q, \varepsilon, \varepsilon) \text{ mit } q \in F\}$
Im Allgemeinen sind die Sprachen verschieden.

Lemma 3.47. $\mathfrak{L}(\Sigma, \mathrm{PDA}, F) = \mathfrak{L}(\Sigma, \mathrm{PDA}, \varepsilon) = \mathfrak{L}(\Sigma, \mathrm{PDA}, F, \varepsilon) =: \mathfrak{L}(\Sigma, \mathrm{PDA})$

3.6.3 Nicht-Determinismus

$\delta(q,a,A) = \{(q_1,\alpha_1),(q_2,\alpha_2)\}$
$\delta(q,\varepsilon,A) = \{(q_1',\alpha_1'),(q_2',\alpha_2')\}$
Folgezustände von $(q,aW,A\beta)$: $\vdash (q_1,w,\alpha_1\beta)$
$\vdash (q_1',aw,\alpha_1'\beta)$
(zwei weitere)

Ziel. Kellersprachen = CFL

Satz 3.48. $\mathrm{CFL}(\Sigma) \subseteq \mathfrak{L}(\Sigma,\mathrm{PDA})$

Beweis. $L = L(G), G = \langle N,\Sigma,P,S\rangle$
Konstruktion eines $\mathfrak{A}_G \in \mathrm{PDA}(\Sigma)$ mit $L(\mathfrak{A}_G,\varepsilon) = L(G)$.
Idee:

(1) Simulation der Linksableitungen auf Keller

(2) Vergleich der Terminalsymbole des Kellers mit der Eingabe

(1a) Richtigen Ableitungsschritt „raten“ (nicht-deterministisch)

$\mathfrak{A}_G = \langle Q,\Sigma,\Gamma,\delta,q_0,F,Z_0\rangle \in \mathrm{PDA}(\Sigma)$ mit $Q := \{q\}, \Gamma := N \cup \Sigma, Z_0 := S$
Vergleichsschritte: $\delta(q,a,a) := \{(q,\varepsilon)\}$ für alle $a \in \Sigma$
Ableitungsschritte: $\delta(q,\varepsilon,A) := \{(q,\beta) | A \to \beta \in P\}$ für alle $A \in N$
Zum Nachweis von $L(G) = L(\mathfrak{A}_G)$ zeigen wir, dass für alle $A \in N$, $w \in \Sigma^*$:

$$A \overset{*}{\Rightarrow}_G w \curvearrowleft (q,w,A) \vdash^* (q,\varepsilon,\varepsilon)$$

Beachte: für alle $v \in \Sigma^*, \alpha \in \mathcal{X}^*$: $(q,wv,A\alpha) \vdash^* (q,v,\alpha)$
Beweis von $(*)$:
„$\curvearrowright$“: Induktion über die Ableitungslänge
$n = 1$: $A \to w \curvearrowright (q,w,A) \vdash (q,w,w) \vdash^* (q,\varepsilon,\varepsilon)$
$n \mapsto n+1$: $A \Rightarrow v_0A_1v_1 \ldots A_rv_r \overset{n}{\Rightarrow} w$
Dann muss gelten: $A_i \overset{n_i}{\Rightarrow} w_i, w = v_0w_1v_1 \ldots w_rv_r$ mit $n_i \leq n$, also gilt dafür die Induktionsvoraussetzung.

$$\begin{aligned}(q,w,A) &\vdash (q,w,v_0A_1v_1 \ldots A_rv_r)\\ &\vdash^* (q,w_1v_1 \ldots w_rv_r,A_1v_1 \ldots A_rv_r)\\ &\underset{\text{IV}}{\vdash^*} (q,v_1w_2 \ldots w_rv_r,v_1A_1 \ldots A_rv_r)\\ &\vdash^* (q,\varepsilon,\varepsilon)\end{aligned}$$

„$\curvearrowleft$“: Induktion über die Länge i der Berechnung
$i = 1$: $(q,w,A) \vdash (q,\varepsilon,\varepsilon) \curvearrowright A \to \varepsilon, w = \varepsilon \curvearrowright A \overset{*}{\Rightarrow} w$
$i \mapsto i+1$: (ähnlich wie für „$\curvearrowright$“) □

Satz 3.49. $\mathfrak{L}(\Sigma, \text{PDA}) \subseteq \text{CFL}(\Sigma)$

Beweis. (siehe Literatur) □

Korollar 3.50. CFL ist unter Schnitt mit regulären Sprachen abgeschlossen.

Beweis. Für $L \in \text{CFL}(\Sigma)$ und $R \in \text{RegL}(\Sigma)$ existieren $\mathfrak{A}_L \in \text{PDA}(\Sigma)$ und $\mathfrak{A}_R \in \text{DFA}(\Sigma)$ mit $L = L(\mathfrak{A}_L, F_L)$ und $R = L(\mathfrak{A}_R)$.
Konstruiere $\mathfrak{A}_L \| \mathfrak{A}_R \in \text{PDA}(\Sigma)$ durch Parallelkonstruktion.
Also:
$Q := \mathfrak{A}_L \times \mathfrak{A}_R$ und $\delta((q_1, q_2), a, Z) \ni ((q_1', q_2'), \alpha) :\curvearrowright \delta_L(q_1, a, Z) \ni (q_1', \alpha)$ und $\delta_R(q_2, a) = q_2'$
$F := F_L \times F_R$ □

Ziel. Der Deklarationszwang von Variablen ist keine kontextfreie Eigenschaft.

Lemma 3.51. $L = \{ww | w \in \{a,b\}^+\} \notin \text{CFL}$

Beweis. Zunächst zeigen wir: $L' = \{a^m b^n a^m b^n | m, n \geq 1\} \notin \text{CFL}$.
Wäre $L' \in CFL$, so ergäbe sich mit dem Pumping-Index k: $a^k b^k a^k b^k = uvwxy$ mit $|\, vwx \,| \leq k$ und $vx \neq \varepsilon$, also auch $uwy \in L'$. Das ist ein Widerspruch, weil die Zahl der vorderen und hinteren a's bzw. b's nicht mehr gleich ist.
$L' = L \cap \{a\}^+\{b\}^+\{a\}^+\{b\}^+$
Wäre L kontextfrei, so auch L'. Widerspruch. □

Korollar 3.52. 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 Substitutionsabbildung mit $h(a) = a, h(b) = b$ und $h(x) = \varepsilon$ sonst.
Dann: $h(L) = \{ww | w \in \{a,b\}^*\} \notin \text{CFL} \curvearrowright L \notin \text{CFL} \curvearrowright P \notin \text{CFL}$. □

3.6.4 Deterministische Kellerautomaten

Definition 3.53 (deterministischer Kellerautomat). $\mathfrak{A} = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$ heißt DETERMINISTISCH, in Zeichen: $\mathfrak{A} \in \text{DPDA}$, wenn gilt:

(i) $|\, \delta(q,a,Z) \,| \leq 1$ für alle $(q,a,Z) \in Q \times \Sigma_\varepsilon \times \Gamma$

(ii) $|\, \delta(q,\varepsilon,Z) \,| = 1 \curvearrowright |\, \delta(q,a,Z) \,| = 0$ für alle $a \in \Sigma$

Folgerung 3.54. Zu jeder Konfiguration (q,w,α) gibt es höchstens eine Folgekonfiguration.

3.6.5 Erkennung durch Endzustände

$L(\mathfrak{A}) := L(\mathfrak{A}, F) = \{w \in \Sigma^* | (q_0, w, Z_0) \vdash^* (q, \varepsilon, \alpha) \text{ mit } q \in F \text{ und } \alpha \in \Gamma^*\}$
Es gilt nicht die Äquivalenz zur Erkennung mit leerem Keller bzw. mit leerem Keller und Endzuständen.
Es gilt:
$\mathfrak{L}(\Sigma, \text{DPDA}, \varepsilon) \subsetneqq \mathfrak{L}(\Sigma, \text{DPDA}, F) =: \mathfrak{L}(\Sigma, \text{DPDA})$ und $\mathfrak{L}(\Sigma, \text{DPDA}, \varepsilon, F) \subsetneqq \mathfrak{L}(\Sigma, \text{DPDA}, F)$.

3.6.6 „Hiobsbotschaft“

Satz 3.55. $\mathfrak{L}(\Sigma, \text{DPDA})$ ist unter Komplement abgeschlossen.

Beweisidee. Zu jedem $\mathfrak{A} \in \text{DPDA}(\Sigma)$ ist ein äquivalenter $\tilde{\mathfrak{A}} \in \text{DPDA}(\Sigma)$ konstruierbar, so dass gilt:
Für jedes $w \in \Sigma^*$ gibt es $q \in \tilde{Q}$ und $\alpha \in \tilde{\Gamma}^*$ mit $(\tilde{q}_0, w, \tilde{Z}_0) \vdash^* (q, \varepsilon, \alpha)$ Endkonfiguration mit $w \in L(\mathfrak{A}) \curvearrowright q \in F$.
Damit ist Komplementbildung wie bei DFAs möglich.
Kern des Beweises:
Elimination von Schleifenkonfiguration, d.h. $(q, w, \alpha) \in Q \times \Sigma^* \times \Gamma^*$ heißt SCHLEIFENKONFIGURATION, falls für jedes $i \in \mathbb{N}$ ein $q_i \in Q$ und ein $\alpha_i \in \Gamma^*$ existieren, so dass $| \alpha_i | \geq | \alpha |$ und $(q, w, \alpha) \vdash^i (q_i, w, \alpha_i)$.
Diese Eigenschaft ist entscheidbar. $\curvearrowright$ Elimation. □

Folgerung 3.56. $\mathfrak{L}(\Sigma, \text{DPDA}) \subsetneqq \mathfrak{L}(\Sigma, \text{PDA}) = \text{CFL}$

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 $G =$ CNF (auf beliebige CFG übertragbar).
Für $G = \langle N, \Sigma, P, S \rangle$, $(A \to BC, A \to a)$ und $w = a_1 \dots a_n$ mit $n > 0$ definieren wir:

$$w_{ij} := a_i \dots a_j \text{ für } 1 \leq i \leq j \leq n$$

und

$$N_{ij} := \{A \in N | A \overset{*}{\Rightarrow}_G w_{ij}\}$$

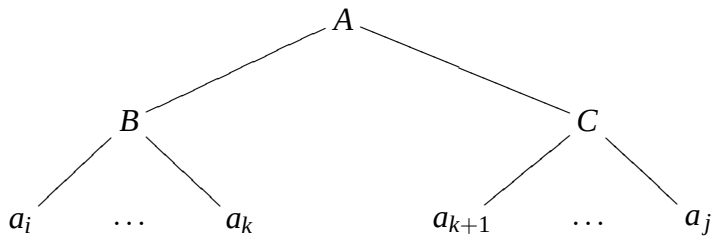
Es gilt:

$$w \in L(G) \curvearrowright S \in N_{1n}$$

Berechne $N_{11}, N_{22}, \dots, N_{nn}$, dann $N_{12}, N_{23}, \dots, N_{n-1,n}$, dann $N_{13}, N_{24}, \dots, N_{n-2,n}$ usw. bis N_{1n}.

(i) $A \in N_{ii} \curvearrowleft A \to a_i \in P$

(ii) $A \in N_{ij}$ mit $i < j \curvearrowleft \exists k, i \leq k < j, \exists A \to BC \in P : B \in N_{ik}$ und $C \in N_{k+1,j}$



3.7.1 Komplexität des Algorithmus

(i) Zeit:

- „for k“: $c \cdot d$
- „for i“: $c \cdot d \cdot (n-d)$
- „for d“: $c \cdot \sum_{d=1}^{n-1} d \cdot (n-d)$

Insgesamt ergibt sich also:

$$
\begin{aligned}
& c \cdot \sum_{d=1}^{n-1} d \cdot (n-d) \\
=& c \cdot \sum_{d=1}^{n} (n \cdot d - d^2) \\
=& c \cdot \left(n \cdot \sum_{d=1}^{n} d - \sum_{d=1}^{n} d^2 \right) \\
=& c \cdot \left(n \cdot \frac{n}{2} \cdot (n+1) - \frac{n \cdot (n+1) \cdot (2n+2)}{6} \right) \\
=& c \cdot \frac{n^3 - n}{6} \\
=& \mathrm{O}(n^3)
\end{aligned}
$$

(ii) Platz: $\mathrm{O}(n^2)$

3.8 Erweiterte kontextfreie Grammatiken (ECFG)

höherer Beschreibungskomfort durch reguläre Ausdrücke als rechte Regelseiten; äquivalente Erweiterung

Definition 3.57 (erweiterte kontextfreie Grammatik). $G = \langle N, \Sigma, P, S \rangle$ ist eine ERWEITERTE KONTEXTFREIE GRAMMATIK, in Zeichen: $G \in \text{ECFG}$, wenn $\langle N, \Sigma, \emptyset, S \rangle \in \text{CFG}$ und $P : N \to \text{RegE}(N \cup \Sigma)$.

Schreibweise 3.58. $A \to \alpha$, falls $P(A) = \alpha$

Semantik. P repräsentiert die Regelmenge $\tilde{P}$ mit $A \to \tilde{\alpha} \in \tilde{P} :\curvearrowright \tilde{\alpha} \in L(P(A))$.

Beachte. $\tilde{P}$ ist im Allgemeinen unendlich.
$L(G) := L(\langle N, \Sigma, \tilde{P}, S \rangle)$

Satz 3.59. $\text{CFL}(\Sigma) = \mathfrak{L}(\Sigma, \text{ECFG})$

Beweisskizze. „$\subseteq$“: nach Def.: $A \to \alpha_1 \vee \alpha_2 \vee \alpha_3$, falls $A \to \alpha_1, A \to \alpha_2, A \to \alpha_3$
„$\supseteq$“: Transformation von ECFG nach CFG
Idee: Elimination regulärer Ausdrücke

$A \to \alpha^*$	ersetzen durch	$A \to \alpha A, A \to \varepsilon$
$A \to \alpha \vee \beta$	ersetzen durch	$A \to \alpha, A \to \beta$
$A \to \alpha \cdot \beta$	ersetzen durch	$A \to BC$ mit neuen B, C: $B \to \alpha, C \to \beta$

□

3.9 Rekursive endliche Automaten (Syntaxdiagramme)

Syntaxdiagramme entsprechen endlichen Automaten, die sich rekursiv aufrufen können.

Neben $(q) \xrightarrow{a} (q')$ ist auch $(q) \xrightarrow{\boxed{r}} (q')$ möglich.

Definition 3.60 (rekursiver endl. Automat). $\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$ heißt REKURSIVER ENDLICHER AUTOMAT über Σ, falls $\delta : Q \times (\Sigma_\varepsilon \cup Q) \to P(Q)$ und ist sonst wie ein endlicher Automat definiert. Bezeichnung: $\mathfrak{A} \in \text{RFA}(\Sigma)$.

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

$$\frac{\delta(q,a) \ni p}{(q,aw) \vdash (p,w)}, \quad \frac{\delta(q,\varepsilon) \ni p}{(q,w) \vdash (p,w)} \text{ und } \frac{\delta(q,r) \ni p, r \in Q \quad (r,w) \vdash^* (s,v), s \in F}{(q,w) \vdash (p,v)}$$

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

Satz 3.61. $\mathfrak{L}(\Sigma, \text{RFA}) = \mathfrak{L}(\Sigma, \text{PDA})$

Beweis. „$\subseteq$“: $\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F\rangle \in \mathrm{RFA}(\Sigma)$
Simulation von $\mathfrak{A}$ durch $\mathfrak{A}' = \langle Q, \Sigma, Q, \delta', q_0, q_0\rangle \in \mathrm{PDA}$
Idee: Keller für „Rücksprungadressen“ (Zustände)
$\delta'(q,a,s) = \{(p,s) | p \in \delta(q,a)\}$ und
$\delta'(q,\varepsilon,s) = \{(p,s) | p \in \delta(q,\varepsilon)\} \cup \{(r,ps) | p \in \delta(q,r)\} \cup \{(s,\varepsilon) | q \in F\}$ für alle $q,r,s,p \in Q$ und $a \in \Sigma$.
„$\supseteq$“: Für $L \in \mathfrak{L}(\Sigma, \mathrm{PDA})$ existiert $G = \langle N, \Sigma, P, S\rangle \in \mathrm{CNF}$ mit $L = L(G)$.
Konstruiere $\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F\rangle \in \mathrm{RFA}$ durch $Q := N \uplus \{q_f\}, q_0 := S, F := \{q_f\}$.
δ: falls $A \to BC$, so $\delta(A,B) \ni C$
$A \to a$, so $\delta(A,a) \ni q_f$
$S \to \varepsilon$, so $\delta(S,\varepsilon) \ni q_f$ □

4 Turingmaschinen und aufzählbare Sprachen

4.1 Chomsky-Grammatiken

Verallgemeinerung kontextfreier Grammatiken
kontextabhängige Ersetzung, Wortersetzung

Definition 4.1 (Chomsky-Grammatik). 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 $G = \langle N, \Sigma, P, S \rangle$ eine CHOMSKY-GRAMMATIK.

Semantik. $\pi = \alpha_1 \to \alpha_2 \in P$ bestimmt die Ableitungsrelation $\Rightarrow_\pi \subseteq \mathcal{X}^* \times \mathcal{X}^*$ durch $\beta_1 \Rightarrow_\pi \beta_2 :\curvearrowright$ es existiert $\gamma, \delta \in \mathcal{X}^*$ mit $\beta_1 = \gamma\alpha_1\delta$ und $\beta_2 = \gamma\alpha_2\delta$.
Ableitungsrelation von G: $\Rightarrow_G := \bigcup_{\pi \in P} \Rightarrow_\pi$
G erzeugt $L(G) = \{w \in \Sigma^* | S \Rightarrow_G^* w\}$.

4.1.1 Klassifikation von Chomsky-Grammatiken und Sprachfamilien

Definition 4.2 (Typ i-Grammatik). Sei $G = \langle N, \Sigma, P, S \rangle$ eine Chomsky-Grammatik und sei $i \in \{0, 1, 2, 3\}$. Dann heißt G VOM TYP i oder eine TYP i-GRAMMATIK, wenn P die Eigenschaft (i) besitzt mit:

(0) keine Einschränkung

(1) Jedes $\pi \in P$ hat die Form $\gamma A \delta \to \gamma\beta\delta$ mit $\beta \neq \varepsilon$ oder $S \to \varepsilon$.
Ferner: $S \to \varepsilon, \alpha \to \beta \in P \curvearrowright S$ nicht in P.

(2) Jedes $\pi \in P$ hat die Form $A \to \alpha$.

(3) Entweder: Jedes $\pi \in P$ hat die Form $A \to Bw$ oder $A \to w$.
Oder: Jedes $\pi \in P$ hat die Form $A \to wB$ oder $A \to w$.

Grammatiken vom Typ 0 sind genau die Chomsky-Grammatiken.
Grammatiken vom Typ 1 heißen auch KONTEXTSENSITIV.
Grammatiken vom Typ 2 sind genau die kontextfreien Grammatiken.
Grammatiken vom Typ 3 sind genau die einseitig-linearen Grammatiken.
$L \subseteq \Sigma^*$ heißt VOM TYP $i \curvearrowright \exists G$ vom Typ $i : L(G) = L$.

Bezeichnung 4.3. $\mathfrak{L}_i(\Sigma) := \{L \subseteq \Sigma^* | L \text{ vom Typ } i\}$

4.1.2 Chomsky-Hierarchie

$\mathfrak{L}_3(\Sigma) \subsetneqq \mathfrak{L}_2(\Sigma) \subsetneqq \mathfrak{L}_1(\Sigma) \subsetneqq \mathfrak{L}_0(\Sigma)$, falls $|\Sigma| > 1$
$|\Sigma| = 1 \curvearrowright \mathfrak{L}_3(\Sigma) = \mathfrak{L}_2(\Sigma)$
Bereits gezeigt:
$\mathfrak{L}_3(\Sigma) = \mathrm{RegL}(\Sigma) \subsetneqq \mathrm{CFL}(\Sigma) = \mathfrak{L}_2(\Sigma)$
außerdem: $\mathfrak{L}_2(\Sigma) \subseteq \mathfrak{L}_1(\Sigma) \subseteq \mathfrak{L}_0(\Sigma)$ (ε-freie CFG bzw. nach Definition)

Definition 4.4 (normierte Grammatik). Sei $G = \langle N, \Sigma, P, S\rangle$ vom Typ 0. 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 $\alpha_2 = \gamma\beta\delta$. Beachte: $\beta = \varepsilon$ ist erlaubt im Gegensatz zu Typ 1.
- Σ-Symbole nur in Regeln der Form $A \to \alpha$.

Satz 4.5. Zu jedem $G = \langle N, \Sigma, P, S\rangle$ vom Typ 0 lässt sich eine äquivalente normierte Grammatik $G' = \langle N', \Sigma, P', S'\rangle$ konstruieren.

Beweis. (1) Terminalsymbol-Bedingung:

$a \in \Sigma \mapsto A_a$ neues N-Symbol

a durch A_a ersetzen

$A_a \to a$ hinzufügen

(2) Simulation von $A_1 \ldots A_n \to B_1 \ldots B_m (n \geq 1, m \geq 0)$ durch Symbolersetzungsregeln: Wähle neue N-Symbole A'_i:

$$
\begin{aligned}
&A_1 \ldots A_n \to A'_1 A_2 \ldots A_n \\
&A'_1 A_2 \ldots A_n \to A'_1 A'_2 A_3 \ldots A_n \\
&\vdots \\
&A'_1 \ldots A'_{n-1} A_n \to A'_1 \ldots A'_n \\
&A'_1 \ldots A'_n \to B_1 A'_2 \ldots A'_n \\
&\vdots
\end{aligned}
$$

Fall 1 ($n \leq m$): $B_1 \ldots B_{n-1} A'_n \to B_1 \ldots B_m$

Fall 2 ($n > m$):

$$
\begin{aligned}
&B_1 \ldots B_m A'_{m+1} \ldots A'_n \to B_1 \ldots B_m A'_{m+2} \ldots A'_n \\
&\vdots \\
&B_1 \ldots B_m A'_n \to B_1 \ldots B_m
\end{aligned}
$$

□

4.1.3 Abschlusseigenschaften von $\mathfrak{L}_0(\Sigma)$

Satz 4.6. $\mathfrak{L}_0(\Sigma)$ ist unter Substitution abgeschlossen, d.h.: ist

$$\varphi : P(\Sigma^*) \to P\left(\left(\bigcup_{a\in\Sigma}\Sigma_a\right)^*\right)$$

eine Substitutionsabbildung, so gilt:

$$L \in \mathfrak{L}_0(\Sigma), \varphi(a) \in \mathfrak{L}_0(\Sigma_a) \text{ für alle } a \in \Sigma \curvearrowright \varphi(L) \in \mathfrak{L}_0\left(\bigcup_{a\in\Sigma}\Sigma_a\right)$$

Beweis. Die Konstruktion von CFG ist nicht anwendbar. Denn: Neue Ableitungen sind nicht möglich wegen unzulässiger Satzformen.
Idee: Sequentialisierung der Ableitungen mit Kontrollsymbol.
$L = L(G)$ mit $G = \langle N, \Sigma, P, S\rangle$ vom Typ 0.
$\varphi(a) =: L_a = L(G_a)$ mit $G_a = \langle N_a, \Sigma_a, P_a, S_a\rangle$ vom Typ 0.
O.B.d.A. sind G, G_a normiert und N, N_a disjunkt.
Konstruktion von $\overline{G}$ mit $L\left(\overline{G}\right) = \varphi(L)$:

$$\overline{N} := N \uplus \left(\biguplus_{a\in\Sigma} N_a\right) \uplus \{A_a | a \in \Sigma\} \uplus \left\{\overline{S}, C\right\}$$

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

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

P_0 entstehe aus P durch Ersetzen von a durch A_a ($a \in \Sigma$).
$P_1 := \{CA_a \to CS_a | a \in \Sigma\}$
$P_2 := \{C_a \to aC | a \in \Sigma\}$ □

Korollar 4.7. $\mathfrak{L}_0(\Sigma)$ ist unter regulären Operationen abgeschlossen:

$$L, L' \in \mathfrak{L}_0(\Sigma) \curvearrowright L \cup L', LL', L^* \in \mathfrak{L}_0(\Sigma)$$

Satz 4.8. $\mathfrak{L}_0(\Sigma)$ ist unter Durchschnitt abgeschlossen.

Beweis. $L_i = L(G_i)$ und $G_i = \langle N_i, \Sigma, P_i, S_i\rangle$ für $i = 1, 2$.
O.B.d.A. sei $N_1 \cap N_2 = \emptyset$.
Konstruktion von $G = \langle N, \Sigma, P, S\rangle$:
$N := N_1 \uplus N_2 \uplus \{A_a | a \in \Sigma\} \uplus \{S, C_1, C_2\}$
$P := P_1 \cup P_2 \cup Q$
$Q := \{S \to C_1S_1C_2S_2C_1, C_2a \to A_aC_2, bA_a \to A_ab, C_1A_aa \to aC_1, C_1C_2C_1 \to \varepsilon | a, b \in \Sigma\}$ □

Bemerkung 4.9. Da $\mathfrak{L}_0(\Sigma) = \mathfrak{L}(\Sigma, \mathrm{TM})$ (siehe unten), ist $\mathfrak{L}_0(\Sigma)$ nicht unter Komplement abgeschlossen.

4.1.4 Kontextsensitive Grammatiken und Sprachen

Definition 4.10 (Grammatik von wachsender Länge). $G = \langle N, \Sigma, P, S\rangle \in \mathrm{CFG}(\Sigma)$ heißt VON WACHSENDER LÄNGE: $\curvearrowright$ für jede Regel $\alpha \to \beta \in P$ gilt: $\mid \alpha \mid \leq \mid \beta \mid$, es sei denn, dass $S \to \varepsilon \in P$ und S auf keiner rechten Regelseite.

Korollar 4.11. Eine Typ 1-Grammatik ist von wachsender Länge.

Satz 4.12. Zu jeder Grammatik von wachsender Länge lässt sich eine äquivalente Typ 1-Grammatik konstruieren.

Beweis. Normierung ohne Fall 2 im Beweis. □

Definition 4.13 (Platzbedarf). Sei $G = \langle N, \Sigma, P, S\rangle$ eine Typ 0-Grammatik, $\delta = (S \Rightarrow \alpha_0 \Rightarrow \alpha_1 \ldots \Rightarrow \alpha_n)$ eine Ableitung von G und $w \in L(G)$. Dann heißt:

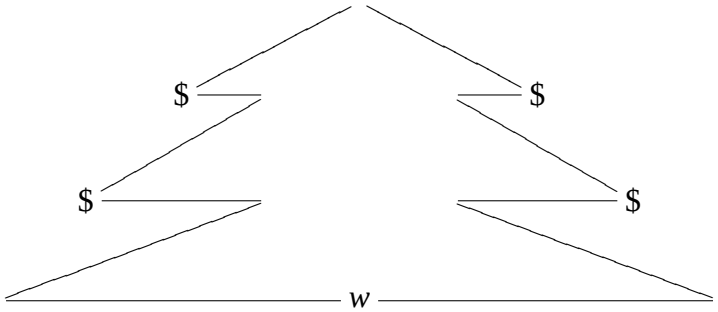
- $\underline{\mathrm{pl}}_G(\delta) := \max\{\mid \alpha_i \mid \mid 0 \leq i \leq n\}$ DER PLATZBEDARF VON δ.
- $\underline{\mathrm{pl}}_G(w) := \min\{\underline{\mathrm{pl}}_G(\delta) \mid \delta = (S \Rightarrow \ldots \Rightarrow w)\}$ DER PLATZBEDARF VON w BZGL. G.

Folgerung 4.14. (1) $\underline{\mathrm{pl}}_G(w) \geq \mid w \mid$

(2) G von wachsender Länge, $w \neq \varepsilon \curvearrowright \underline{\mathrm{pl}}_G(w) = \mid w \mid$

Satz 4.15 (Platzbedarfssatz). Sei G vom Typ 0 und $p \in \mathbb{N}$, so dass für alle $w \in L(G) \setminus \{\varepsilon\}$: $\underline{\mathrm{pl}}_G(w) \leq p \cdot \mid w \mid$.
Dann ist $L(G) \in \mathfrak{L}_1$.

Beweis. Sei $p = 1$, also für $w \in L(G) \setminus \{\varepsilon\}$: $\underline{\mathrm{pl}}_G(w) = \{w\}$
Idee:



(1) $\alpha \to \beta \in P$ mit $\mid \alpha \mid = \mid \beta \mid + i$ $(i > 0) \curvearrowright \alpha \to \$^i\beta$

(2) $\alpha \to \beta \in P$ mit $\mid \alpha \mid \leq \mid \beta \mid \curvearrowright \$^j\alpha \to \beta \in P'$ für alle $j = 0, 1, \ldots, \mid \beta \mid - \mid \alpha \mid$

Für $p > 1$: Induktion. Idee: $N' \supseteq N^P$. □

4.1.5 Abschlusseigenschaften von $\mathfrak{L}_1(\Sigma)$ mit Hilfe des Platzbedarfssatzes

Satz 4.16. $\mathfrak{L}_1(\Sigma)$ ist unter ε-freier Substitution ($\varepsilon \notin \varphi(a)$) abgeschlossen.

Beweis. Platzbedarf der Ausgangssprache mit $p = 1 \curvearrowright$ Platzbedarf bei Substitutionsgrammatik: $\mid w \mid +1 \leq 2 \cdot \mid w \mid$ wegen zusätzlichem Kontrollsymbol C. □

Korollar 4.17. $\mathfrak{L}_1(\Sigma)$ ist unter regulären Operationen abgeschlossen.

Korollar 4.18. $\mathfrak{L}_1(\Sigma)$ 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-Grammatiken hat dann einen Platzbedarf von $2 \cdot \mid w \mid +3 \leq 5 \cdot \mid w \mid$. □

Korollar 4.19. $\mathfrak{L}_2(\Sigma) \subsetneqq \mathfrak{L}_1(\Sigma)$

Ein lange ungelöstes Problem: Abschluss von $\mathfrak{L}_1$ unter Komplement (1987).

4.2 Turingmaschinen, linear beschränkte Automaten

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

Definition 4.20 ((nicht-deterministisch) erkennende Turingmaschine).
$\mathfrak{A} = \langle Q, \Sigma, \Gamma, q_0, \square, F, \delta \rangle \in \mathrm{TM}$, wenn Q Zustandsmenge, Σ Eingabealphabet, Γ Arbeitsalphabet mit $\Sigma \subseteq \Gamma$, $q_0 \in Q$ Anfangszustand, $\square \in \Gamma \setminus \Sigma$ Blank, $F \subseteq Q$ Endzustandsmenge und $\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, N, R\})$ Transitionsfunktion.
$\mathfrak{A}$ heißt (NICHT-DETERMINISTISCH) ERKENNENDE TURINGMASCHINE.

Schreibweise 4.21. Falls $\delta(q, a) \ni (q', b, L)$, so schreibt man $qaq'bL$ (Quintupel).

Semantik. Konfigurationsmenge: $Q \times \Gamma^* \times \Gamma \times \Gamma^* =: \underline{\mathrm{Conf}}(\mathfrak{A})$
Anfangskonfiguration von $w \in \Sigma^*$:

$$\kappa_w = \begin{cases} (q_0, \varepsilon, a, w') & , w = aw' \\ (q_0, \varepsilon, \square, \varepsilon) & , w = \varepsilon \end{cases}$$

Beispiel für Folgekonfiguration: $\delta(q_0, a) \ni (q_1, b, L)$
Endkonfiguration: (q, α, X, β), falls $q \in F$
Die von $\mathfrak{A}$ erkannte Sprache:

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

Satz 4.22. $\mathfrak{L}(\Sigma, \mathrm{TM}) = \mathfrak{L}_0(\Sigma)$

Linear beschränkte TM: TM mit Platzbedarf mit $p\cdot \mid w \mid$
LBA: TM mit Platzbedarf mit $\mid w \mid$

Satz 4.23. $\mathfrak{L}_1(\Sigma) = \mathfrak{L}(\Sigma, \mathrm{LBA})$

Definition 4.24 (deterministische Turingmaschine). $\mathfrak{A} \in \mathrm{TM}(\Sigma)$ ist DETERMINISTISCH: $\curvearrowright$ $\curvearrowleft$ $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, N, R\}$

Satz 4.25. Zu jedem $\mathfrak{A} \in \mathrm{TM}(\Sigma)$ existiert ein äquivalentes $\mathfrak{A}' \in \mathrm{det.TM}(\Sigma) : L(\mathfrak{A}) = L(\mathfrak{A}')$.

4.3 Aufzählbare und entscheidbare Sprachen

Intuitiv: $L \subseteq \Sigma^*$ aufzählbar, wenn es ein effektives (algorithmisches) Verfahren gibt, welches genau die Elemente von L erzeugt. (rekursiv aufzählbar)
Präzisierung durch Turingmaschine mit Ausgabe:
$\mathfrak{A} = \langle Q, \Sigma, \Gamma, q_0, F, \square, \delta \rangle \in \mathrm{DTM}(\Sigma)$
$\delta : Q \times \Gamma \dashrightarrow Q \times \Gamma \times \{L, N, R\}$

Schreibweise 4.26 (für Konfigurationsmenge). Statt (q, α, X, β) einfach $\alpha q X \beta$.

$\mathfrak{A}$ berechnet $f_{\mathfrak{A}} : \Sigma^* \dashrightarrow \Sigma^*$ mit $f_{\mathfrak{A}}(w) = v : \curvearrowright q_0 w \square \vdash^* \alpha q v \square \beta \nvdash$
$\kappa \nvdash$ heißt: κ hat keine Folgekonfigurationen.

4.3.1 Aufzählbare Sprachen

Definition 4.27 (aufzählbar). $L \subseteq \Sigma^*$ heißt AUFZÄHLBAR, wenn $\mathfrak{A} \in \mathrm{DTM}(\Sigma)$ existiert mit $\mathrm{Def}(f_{\mathfrak{A}}) = \Sigma^*$ und $\mathrm{Bild}(f_{\mathfrak{A}}) = L$ oder wenn $L = \emptyset$.

Satz 4.28. Für $L \subseteq \Sigma^*$ gilt:
L aufzählbar $\curvearrowleft L \in \mathfrak{L}(\Sigma, \mathrm{TM})$

Beweisskizze. Sei zunächst $L \neq \emptyset$.
„$\curvearrowright$“: $\mathfrak{A} \in \mathrm{DTM}(\Sigma)$ mit $f_{\mathfrak{A}}$ total und $L = \mathrm{Bild}(f_{\mathfrak{A}})$.
Konstruktion von $\mathfrak{A}' \in \mathrm{DTM}(\Sigma)$ mit $L(\mathfrak{A}') = L$.
Idee: Eingabe $w \in \Sigma^*$ speichern, L aufzählbar und jedes aufgezählte Wort mit w vergleichen; bei Übereinstimmung Endzustand.
„$\curvearrowleft$“: $\mathfrak{A} \in \mathrm{DTM}(\Sigma)$ mit $L(\mathfrak{A}) = L$.
Konstruktion von $\mathfrak{A}'' \in \mathrm{DTM}$ mit $\mathrm{Bild}(f_{\mathfrak{A}''}) = L$ und $\mathrm{Def}(f_{\mathfrak{A}''}) = \Sigma^*$.
Idee:

(1) $\mathfrak{A}$ so modifizieren, dass Eingabe bei Erkennung ausgegeben wird.

Dann: $\mathrm{Def}(f_{\mathfrak{A}'}) = \mathrm{Bild}(f_{\mathfrak{A}'}) = L$

(2) Damit $\mathfrak{A}'$ bei jeder Eingabe ein Wort aus L ausgibt, folgender Trick:

Teil der Eingabe als Zähler zur Beschränkung der Rechenlänge $\Sigma = \{a_1, \ldots, a_r\}$ mit $r \geq 2$.

Eingabe hat immer die Form $a_1{}^n a_i w$ bzw. $a_1{}^n$ mit $i \neq 1$ und $n \in \mathbb{N}$.

$\mathfrak{A}''$ simuliert dann $\mathfrak{A}'$ mit Eingabe von w bzw. ε und höchstens n Schritten. Eine Ausgabe von $\mathfrak{A}'$ ist auch Ausgabe von $\mathfrak{A}''$.

Wurde nach n Schritten keine Ausgabe berechnet, so wird irgendein $v \in L$ als Ausgabe berechnet:

$\mathfrak{A}'$ mit Längenbeschränkung und Eingabe von $\varepsilon, a_1, a_2, \ldots$ bis zur ersten Ausgabe laufen lassen.

Es folgt: $f_{\mathfrak{A}''}$ total mit Bild$(f_{\mathfrak{A}}) = L$.

Für $L = \emptyset$ gilt nach Definition: L aufzählbar.
$\emptyset$ ist auch TM-erkennbar: q_1aaNa_0 ($\{q_1\} \in F$), q_0bbNq_0 □

Satz 4.29. $\mathfrak{L}_0(\Sigma) \subsetneq P(\Sigma^*)$

Beweis. Die Menge der Typ 0-Grammatiken über Σ lässt sich nach ihrer Größe abzählen. Also: $\mathfrak{L}_0$ abzählbar, d.h. gleichmächtig zu $\mathbb{N}$. □

$P(\Sigma^*)$ ist jedoch überabzählbar.

4.3.2 Diagonalverfahren nach Cantor

$|\Sigma| = 1, \Sigma^* = \mathbb{N}, L \subseteq \Sigma^* \mapsto \underline{\text{char}}_L : \mathbb{N} \to \{0,1\}$, $n \in L \curvearrowright \underline{\text{char}}_L(n) = 1$
Angenommen, $P(\mathbb{N})$ wäre abzählbar. Dann gäbe es $f : \mathbb{N}^2 \to \{0,1\}$.

	0	1	2	3	4	…
0	0	1	0	0	1	…
1	1	0	1	0	0	…
2	0	1	1			…
⋮				⋱		

Definiere $L_{\text{dia}} \subseteq \mathbb{N}$ durch $\underline{\text{char}}_{L_{\text{dia}}}(j) = 1 - f(j,j)$. Dann kann L_{dia} nicht in der Abzählung vorkommen.

4.3.3 Entscheidbare Sprachen

Intuitiv: $L \subseteq \Sigma^*$ entscheidbar, wenn es ein effektives Verfahren gibt, welches für jedes $w \in \Sigma^*$ feststellt, ob $w \in L$ oder ob $w \notin L$ gilt.
Präzisierung durch TM mit Ausgabe.

Definition 4.30 (entscheidbar).
$L \subseteq \Sigma^*$ heißt ENTSCHEIDBAR: $\curvearrowright \exists\, \mathfrak{A} \in \mathrm{DTM}(\Sigma)$ mit $\mathrm{Def}(f_{\mathfrak{A}}) = \Sigma^*$ und $L = {f_{\mathfrak{A}}}^{-1}(\varepsilon)$

Satz 4.31. Für $L \subseteq \Sigma^*$ gilt:
L entscheidbar $\curvearrowright L$ und $\Sigma^* \setminus L$ aufzählbar.

Beweis. „$\curvearrowright$“: L entscheidbar: $\mathfrak{A} \in \mathrm{DTM}(\Sigma)$ mit $f_{\mathfrak{A}}$ total und $L = {f_{\mathfrak{A}}}^{-1}(\varepsilon)$
Konstruiere $\mathfrak{A}'$ mit $L(\mathfrak{A}') = L$ und $\mathfrak{A}''$ mit $L(\mathfrak{A}'') = \Sigma^* \setminus L$ durch passende Wahl der erkennenden Endzustände.
„$\curvearrowleft$“: Reißverschlussverfahren zweier Aufzählungsmaschinen □

5 Unentscheidbare Probleme

Bekannt:

- Entscheidbar (ATFS):
  - Wortproblem für Typ 1-/2-/3-Grammatiken („$w \in L(G)$?“)
- Unentscheidbar (BuK):
  - Halteproblem für TM („Hält $\mathfrak{A}$ auf Eingabe w?)
  - Postsches Korrespondenzproblem (PCP)

Ziel. Unentscheidbarkeit

- des Schnittproblems für (D)PDA
- des Äquivalenzproblems für CFG

Satz 5.1 (Schnittproblem für CFG). Es ist nicht entscheidbar, ob für $G_1, G_2 \in \text{CFG}$ gilt:

$$L(G_1) \cap L(G_2) = \emptyset$$

Beweis (durch Reduktion des PCP auf das Schnittproblem).
Seien $v = (v_1, \ldots, v_n), w = (w_1, \ldots, w_n) \in (\Sigma^*)^n$.
Wir konstruieren $G_v, G_w \in \text{CFG}$, so dass $\text{PCP}(v,w)$ lösbar $\curvearrowleft L(G_1) \cap L(G_2) = \emptyset$.
Sei dazu $\tilde{\Sigma} := \Sigma \cup \{a_1, \ldots, a_n\}$ und

$$G_v = \left\langle \{S_v\}, \tilde{\Sigma}, P_v, S_v \right\rangle \text{ mit den Produktionen } P_v := \bigcup_{i=1}^{n} \{S_v \to a_i S_v v_i \mid a_i v_i\}$$

(G_w analog).
Es folgt:

$$L_v := L(G_v) = \{a_{i_1} \ldots a_{i_k} v_{i_k} \ldots v_{i_1} | i \geq 1, 1 \leq j \leq n\}$$

(L_w analog)
Und somit:

$$\begin{aligned} L_v \cap L_w \neq \emptyset &\curvearrowleft \text{es ex. } k \geq 1,\ i_1, \ldots, i_k \in \{1, \ldots, n\} \\ &\qquad \text{mit } a_{i_1} \ldots a_{i_k} v_{i_k} \ldots v_{i_1} = a_{i_1} \ldots a_{i_k} w_{i_k} \ldots w_{i_1} \\ &\curvearrowleft \text{es ex. } k \geq 1,\ i_1, \ldots, i_k \in \{1, \ldots, n\} \\ &\qquad \text{mit } v_{i_k} \ldots v_{i_1} = w_{i_k} \ldots w_{i_1} \\ &\curvearrowleft \text{PCP}(v,w) \text{ lösbar} \end{aligned}$$

Die Entscheidbarkeit des Schnittproblems würde aber die Entscheidbarkeit des PCP implizieren. □

Korollar 5.2. Das Schnittproblem für DPDA ist nicht entscheidbar.

Beweis. L_v und L_w sind offensichtlich deterministisch kontextfrei. □

Satz 5.3 (Äquivalenzproblem für CFG). Es ist nicht entscheibar, ob für $G_1, G_2 \in \text{CFG}$ gilt:

$$L(G_1) = L(G_2)$$

Beweis (durch Reduktion des Schnittproblems für DPDA auf das Äquivalenzproblem für CFG).
(1) Zu $\mathfrak{A} \in \text{DPDA}(\Sigma)$ lässt sich $\overline{\mathfrak{A}} \in \text{DPDA}(\Sigma)$ mit $L(\overline{\mathfrak{A}}) = \overline{L(\mathfrak{A})}$ konstruieren.

(2) Zu einem $\mathfrak{A} \in \text{PDA}(\Sigma)$ lässt sich $G_{\mathfrak{A}}$ konstruieren mit $L(G_{\mathfrak{A}}) = L(\mathfrak{A})$.

(3) Zu $G_1, G_2 \in \text{CFG}(\Sigma)$ lässt sich $G_\cup \in \text{CFG}(\Sigma)$ konstruieren mit $L(G_\cup) = L(G_1) \cup L(G_2)$.
Damit folgt für $\mathfrak{A}_1, \mathfrak{A}_2 \in \text{DPDA}$:

$$\begin{aligned}
L(\mathfrak{A}_1) \cap L(\mathfrak{A}_2) = \emptyset &\curvearrowright L(\mathfrak{A}_1) \subseteq \overline{L(\mathfrak{A}_2)} \\
&\underset{(1)}{\curvearrowright} L(\mathfrak{A}_1) \subseteq L\left(\overline{\mathfrak{A}_2}\right) \\
&\curvearrowright L(\mathfrak{A}_1) \cup L\left(\overline{\mathfrak{A}_2}\right) = L\left(\overline{\mathfrak{A}_2}\right) \\
&\underset{(2)}{\curvearrowright} L(G_{\mathfrak{A}_1}) \cup L(G_{\overline{\mathfrak{A}_2}}) = L(G_{\overline{\mathfrak{A}_2}}) \\
&\underset{(3)}{\curvearrowright} L(G_\cup) = L(G_{\overline{\mathfrak{A}_2}})
\end{aligned}$$

Die Entscheidbarkeit des Äquivalenzproblems würde somit die Entscheidbarkeit des Schnittproblems implizieren. □

Übersicht: Definitionen

Index