

Automaten und Sprachen

§ 7–10: Grammatiken und kontextfreie Sprachen



Teil I: Endliche Automaten und reguläre Sprachen

0. Grundbegriffe
1. Endliche Automaten
2. Nachweis der Nichterkennbarkeit
3. Abschlusseigenschaften
4. Entscheidungsprobleme
5. Reguläre Ausdrücke und Sprachen
6. Minimale DEAs und die Nerode-Rechtskongruenz

Teil II: Grammatiken, kontextfreie Sprachen und Kellerautomaten

- 
7. Die Chomsky-Hierarchie
 8. Rechtslineare Grammatiken und reguläre Sprachen
 9. Normalformen und Entscheidungsprobleme
 10. Abschlusseigenschaften und Pumping-Lemma
 11. Kellerautomaten
 12. Die Struktur kontextfreier Sprachen



§7 Die Chomsky-Hierarchie



Grammatiken

Grammatiken sind neben Automaten und regulären Ausdrücken ein weiteres wichtiges Mittel, um formale Sprachen zu definieren:

- Man beginnt mit einem Startsymbol.
- Regeln erlauben es, wiederholt (Teil-)Wörter durch andere Wörter zu ersetzen.

Beispiel 7.1

Regeln:

$$S \rightarrow aSb \quad (1)$$

$$S \rightarrow \varepsilon \quad (2)$$

Startsymbol: S

$$\begin{array}{c} S \stackrel{(1)}{\vdash} aSb \stackrel{(1)}{\vdash} aaSbb \\ \qquad\qquad\qquad \stackrel{(1)}{\vdash} aaaSbbb \stackrel{(2)}{\vdash} aaabbb \end{array}$$

Das Symbol S ist Hilfssymbol (**nichtterminales** Symbol).

Man ist nur an erzeugten **Terminalwörtern** interessiert (**ohne** Hilfssymbol).

In diesem Fall sind das die Wörter $a^n b^n$ mit $n \geq 0$.



Grammatiken

Definition 7.2 (Grammatik)

Eine **Grammatik** ist von der Form $G = (N, \Sigma, P, S)$, wobei

- N und Σ endliche, disjunkte Alphabete sind
(N : Nichtterminalsymbole, Σ : Terminalsymbole),
- $S \in N$ das Startsymbol ist,
- $P \subseteq (N \cup \Sigma)^+ \times (N \cup \Sigma)^*$ eine endliche Menge von Ersetzungsregeln (Produktionen) ist.

Produktionen $(u, v) \in P$ schreibt man gewöhnlich als $u \rightarrow v$.

Beispiel 7.3

$G = (N, \Sigma, P, S)$ mit $N = \{S, B\}$, $\Sigma = \{a, b, c\}$ und

$$P = \{S \rightarrow aSBc, S \rightarrow abc, cB \rightarrow Bc, bB \rightarrow bb\}$$

Wir schreiben Elemente von N mit Groß- und von Σ mit Kleinbuchstaben.



Ableitbarkeit, erzeugte Sprache

Definition 7.4 (Ableitbarkeit, erzeugte Sprache)

Es sei $G = (N, \Sigma, P, S)$ eine Grammatik und $x, y \in (N \cup \Sigma)^*$.

1. y aus x direkt ableitbar:

$x \vdash_G y$ gdw.

$x = x_1 u x_2$ und $y = x_1 v x_2$ mit $u \longrightarrow v \in P$ und $x_1, x_2 \in (N \cup \Sigma)^*$

2. y aus x in n Schritten ableitbar:

$x \vdash_G^n y$ gdw. $x \vdash_G x_1 \vdash_G \cdots \vdash_G x_{n-1} \vdash_G y$ für $x_1, \dots, x_{n-1} \in (N \cup \Sigma)^*$

3. y aus x ableitbar:

$x \vdash_G^* y$ gdw. $x \vdash_G^n y$ für ein $n \geq 0$

4. Die durch G erzeugte Sprache:

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



Beispiel

Beispiel 7.5

$G = (N, \Sigma, P, S)$ mit $N = \{S, B\}$, $\Sigma = \{a, b\}$ und

$$P = \left\{ \begin{array}{l} S \rightarrow aS, \\ S \rightarrow bS, \\ S \rightarrow abB, \\ B \rightarrow aB, \\ B \rightarrow bB, \\ B \rightarrow \varepsilon \end{array} \right\}$$

$$L(G) = \underline{\Sigma^*} \cdot \underline{\{a\}} \cdot \underline{\{b\}} \cdot \underline{\Sigma^*}$$



Beispiel

Beispiel 7.3 (Fortsetzung)

$G = (N, \Sigma, P, S)$ mit $N = \{S, B\}$, $\Sigma = \{a, b, c\}$ und

$$P = \{S \rightarrow aSBc, S \rightarrow abc, cB \rightarrow Bc, bB \rightarrow bb\}$$

$S \vdash_G abc$, d. h. $abc \in L(G)$

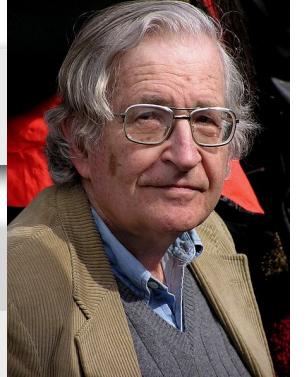
$S \vdash_G a\overline{S}Bc$ $\vdash_G aa\overline{S}BcBc$ $\vdash_G aaabc\overline{B}cBc$ $\vdash_G aaab\overline{B}ccBc$

$\vdash_G [2] aaab\overline{B}Bccc$ $\vdash_G [2] aaabb\overline{B}ccc$, d. h. $aaabbccc \in L(G)$

Es gilt: $L(G) = \{a^n b^n c^n \mid n \geq 1\}$ (Tafel)



Chomsky-Hierarchie



Definition 7.6 (Chomsky-Hierarchie, Typen von Grammatiken)

Eine Grammatik $G = (N, \Sigma, P, S)$ ist vom ...

Noam Chomsky
CC BY 2.0

(Duncan Rawlinson)

Typ 0 Jede Grammatik

Typ 1 Monotone Grammatik:

Regeln $w \rightarrow u$ mit $w, u \in (\Sigma \cup N)^+$ und $|u| \geq |w|$ (nicht verkürzend)

Ausnahme: Regel $S \rightarrow \varepsilon$ erlaubt,
wenn S in keiner Produktion auf der rechten Seite vorkommt.

Typ 2 Kontextfreie Grammatik:

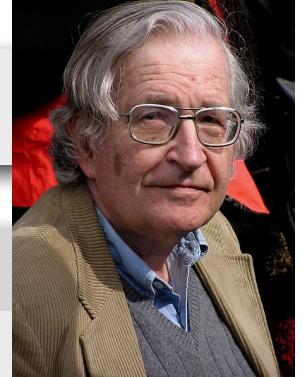
Regeln $A \rightarrow w$ mit $A \in N$, $w \in (\Sigma \cup N)^*$

Typ 3 Rechtslineare Grammatik:

Regeln $A \rightarrow uB$ oder $A \rightarrow u$ mit $A, B \in N$, $u \in \Sigma^*$



Chomsky-Hierarchie



Definition 7.6 (Chomsky-Hierarchie, Typen von Grammatiken)

Eine Grammatik $G = (N, \Sigma, P, S)$ ist vom ...

Noam Chomsky
CC BY 2.0
(Duncan Rawlinson)

Typ 0 Jede Grammatik

Implikationen zwischen Grammatiktypen

Typ 1 Monotone Grammatik:

Regeln $w \rightarrow u$ mit $w, u \in (\Sigma \cup N)^+$ und $|u| \geq |w|$ (nicht verkürzend)



(#) Ausnahme: Regel $S \rightarrow \varepsilon$ erlaubt,
wenn S in keiner Produktion auf der rechten Seite vorkommt.

Typ 2 Kontextfreie Grammatik:

(#) wir kommen später
darauf zurück

Regeln $A \rightarrow w$ mit $A \in N$, $w \in (\Sigma \cup N)^*$

Typ 3 Rechtslineare Grammatik:

Regeln $A \rightarrow uB$ oder $A \rightarrow u$ mit $A, B \in N$, $u \in \Sigma^*$



„Kontextfrei“ versus „kontextsensitiv“

Warum heißt Typ 2 **kontextfrei**?

In kontextfreier Regel

$$A \longrightarrow w$$

besteht linke Seite nur aus Nichtterminalsymbol A ;
dessen Ersetzung ist daher **unabhängig vom Kontext im Wort!**

Grammatiken der **Typen 0 und 1** erlauben auch Regeln der Form

$$u_1 A u_2 \longrightarrow u_1 w u_2$$

Ersetzung von A durch w ist also **abhängig vom Kontext (u_1 links, u_2 rechts)**.

Ursprünglich wurden monotone Grammatiken sogar
durch Regeln dieser Art definiert („kontextsensitive Grammatiken“).



Chomsky-Typen der Beispiel-Grammatiken

Beispiel 7.7 (Beispiele 7.1, 7.3, 7.5 fortgesetzt)

$$\begin{aligned}S &\rightarrow aSb \\S &\rightarrow \varepsilon\end{aligned}$$

Typ 2 (kontextfrei)

$$\begin{aligned}S &\rightarrow aS \\S &\rightarrow bS \\S &\rightarrow abB \\B &\rightarrow aB \\B &\rightarrow bB \\B &\rightarrow \varepsilon\end{aligned}$$

Typ 3 (rechtslinear)

$$\begin{aligned}S &\rightarrow aSBc \\S &\rightarrow abc \\cB &\rightarrow Bc \\bB &\rightarrow bb\end{aligned}$$

Typ 1 (monoton)



Sprachklassen der Chomsky-Hierarchie

Definition 7.8 (Sprachklassen)

Für $i = 0, 1, 2, 3$ ist die **Klasse der Typ- i -Sprachen** definiert als:

$$\mathcal{L}_i := \{L(G) \mid G \text{ ist Grammatik vom Typ } i\}$$

Man bezeichnet die Sprachklassen oft analog zu den Grammatiken, z.B. werden Typ-2-Sprachen auch **kontextfreie Sprachen** genannt.

Er gibt jedoch eine Ausnahme:

Typ-1-Sprachen nennt man i.d.R. **kontextsensitive Sprachen**, nicht **monotone** Sprachen.



Sprachklassen der Chomsky-Hierarchie

Definition 7.8 (Sprachklassen)

Für $i = 0, 1, 2, 3$ ist die **Klasse der Typ- i -Sprachen** definiert als:

$$\mathcal{L}_i := \{L(G) \mid G \text{ ist Grammatik vom Typ } i\}$$

Im Gegensatz zu den Grammatiktypen bilden die Sprachtypen eine **Hierarchie**. Diese ist sogar **strikt**.

Satz 7.9

$$\mathcal{L}_3 \subset \mathcal{L}_2 \subset \mathcal{L}_1 \subset \mathcal{L}_0$$

Beweis.

Nach Definition der Grammatiktypen gilt offenbar $\mathcal{L}_3 \subseteq \mathcal{L}_2$ und $\mathcal{L}_1 \subseteq \mathcal{L}_0$.

Die Inklusion $\mathcal{L}_2 \subseteq \mathcal{L}_1$ zeigen wir später (Satz 9.11); ebenso die Striktheit.



Grammatiktypen: Einordnung

Typ 0: **allgemeine Grammatiken**

gehören in die Theorie der Berechenbarkeit (☞ VL Berechenbark.)

Typ 1: **monotone Grammatiken**

ebenfalls in Berechenbarkeit behandelt

Typ 2: **kontextfreie Grammatiken**

Grundlage für die **Syntax von Programmiersprachen**,
Verwendung (in erweiterter Form) im **Compilerbau (Parser)**

Typ 3: **rechtslineare Grammatiken**

Wir zeigen als nächstes: **erzeugen genau die regulären Sprachen**



Teil I: Endliche Automaten und reguläre Sprachen

0. Grundbegriffe
1. Endliche Automaten
2. Nachweis der Nichterkennbarkeit
3. Abschlusseigenschaften
4. Entscheidungsprobleme
5. Reguläre Ausdrücke und Sprachen
6. Minimale DEAs und die Nerode-Rechtskongruenz

Teil II: Grammatiken, kontextfreie Sprachen und Kellerautomaten

- 
7. Die Chomsky-Hierarchie
 8. Rechtslineare Grammatiken und reguläre Sprachen
 9. Normalformen und Entscheidungsprobleme
 10. Abschlusseigenschaften und Pumping-Lemma
 11. Kellerautomaten
 12. Die Struktur kontextfreier Sprachen



§8 Rechtslineare Grammatiken und reguläre Sprachen



Rechtslinear = regulär

Satz 8.1 (Typ 3 = regulär)

Die Typ-3-Sprachen sind genau die regulären/erkennbaren Sprachen:

$$\mathcal{L}_3 = \{L \mid L \text{ ist regulär}\}$$

Beweis.

1. Jede Typ-3-Sprache ist regulär.
2. Jede reguläre Sprache ist eine Typ-3-Sprache.



Richtung 1: Typ 3 \Rightarrow regulär

Sei $L \in \mathcal{L}_3$, d. h. $L = L(G)$ für eine Typ-3-Grammatik $G = (N, \Sigma, P, S)$.

Zur Erinnerung:

rechtslinear = nur Regeln $A \rightarrow uB$ oder $A \rightarrow u$

$w \in L(G)$ gdw. es Zerlegung $w = w_1 \cdots w_n$ gibt und Ableitung der Form

$$\begin{array}{ccccccccc} B_0 & \vdash_G & w_1 B_1 & \vdash_G & w_1 w_2 B_2 & \vdash_G & \cdots & \vdash_G & w_1 \cdots w_{n-1} B_{n-1} \vdash_G w_1 \cdots w_{n-1} w_n \\ \parallel & & & & & & & & \parallel \\ S & & & & & & & & w \end{array}$$

Beobachtung:

- Nichtterminale verhalten sich wie Zustände eines Automaten (zu jedem Zeitpunkt genau eines).
- Produktionen verhalten sich wie Zustandsübergänge.



Richtung 1: Typ 3 \Rightarrow regulär

Sei $L \in \mathcal{L}_3$, d. h. $L = L(G)$ für eine Typ-3-Grammatik $G = (N, \Sigma, P, S)$.

Wir konstruieren einen **NEA mit Wortübergängen** für L :

$$\mathcal{A} = (N \cup \{\Omega\}, \Sigma, S, \Delta, \{\Omega\}),$$

wobei $\Omega \notin N$ akzeptierender Zustand ist und

$$\begin{aligned}\Delta = \{(A, w, B) \mid A \longrightarrow wB \in P\} \cup \\ \{(A, w, \Omega) \mid A \longrightarrow w \in P\}.\end{aligned}$$

Es gilt $w \in L(G)$

gdw. es gibt Zerlegung $w = w_1 \cdots w_n$ und Ableitung in G

$$S = B_0 \vdash_G w_1 B_1 \vdash_G \cdots \vdash_G w_1 \cdots w_{n-1} B_{n-1} \vdash_G w_1 \cdots w_{n-1} w_n$$

gdw. es gibt Zerlegung $w = w_1 \cdots w_n$ und Pfad in \mathcal{A}

$$S \xrightarrow{w_1} B_1 \xrightarrow{w_2} \cdots \xrightarrow{w_{n-1}} B_{n-1} \xrightarrow{w_n} \Omega$$

gdw. $w \in L(\mathcal{A})$

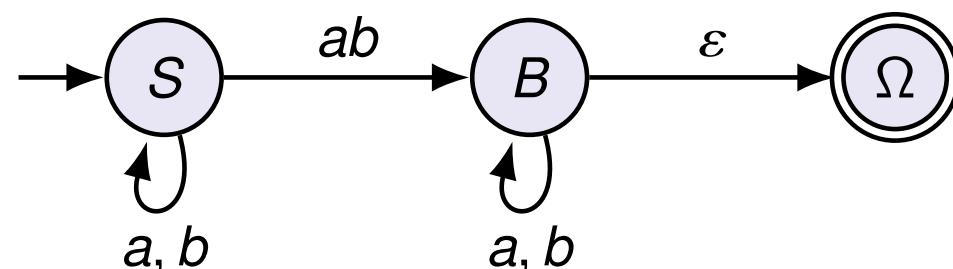
Dies zeigt $L(\mathcal{A}) = L(G)$.



Richtung 1: Typ 3 \Rightarrow regulär

Beispiel 7.5 (Fortsetzung)

$S \rightarrow aS$
 $S \rightarrow bS$
 $S \rightarrow abB$
 $B \rightarrow aB$
 $B \rightarrow bB$
 $B \rightarrow \varepsilon$



Richtung 2: regulär \Rightarrow Typ 3

Es sei $L = L(\mathcal{A})$ für einen NEA $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$.

Wir definieren daraus eine Typ-3-Grammatik $G = (N, \Sigma, P, S)$ wie folgt:

$$N := Q$$

$$S := q_0$$

$$P := \{p \rightarrow aq \mid (p, a, q) \in \Delta\} \cup \{p \rightarrow \varepsilon \mid p \in F\}$$

Ein Pfad in \mathcal{A} der Form

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \cdots \xrightarrow{a_n} q_n \quad \text{mit } q_n \in F$$

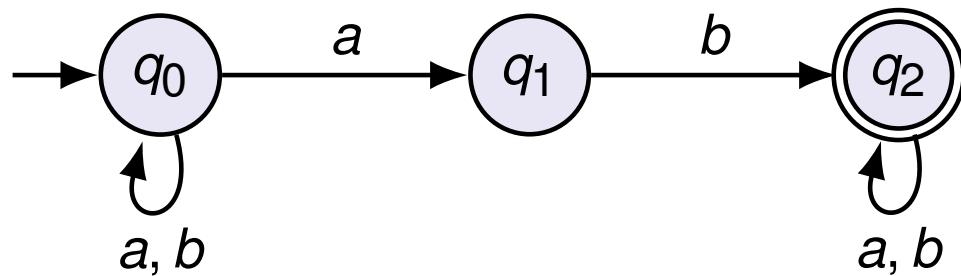
entspricht nun genau einer Ableitung in G :

$$q_0 \vdash_G a_1 q_1 \vdash_G a_1 a_2 q_2 \vdash_G \cdots \vdash_G a_1 \cdots a_n q_n \vdash_G a_1 \cdots a_n$$

Dies zeigt $L(G) = L(\mathcal{A})$. □



Beispiel 8.2



liefert die folgenden
rechtslinearen Produktionen:

$$\begin{array}{ll} q_0 & \longrightarrow aq_0 \\ q_0 & \longrightarrow bq_0 \\ q_0 & \longrightarrow aq_1 \\ q_1 & \longrightarrow bq_2 \\ q_2 & \longrightarrow aq_2 \\ q_2 & \longrightarrow bq_2 \\ q_2 & \longrightarrow \varepsilon \end{array}$$

Regulär vs. kontextfrei

Korollar 8.3

$$\mathcal{L}_3 \subset \mathcal{L}_2$$

Beweis.

Wir wissen bereits: $\mathcal{L}_3 \subseteq \mathcal{L}_2$

Wieso ist die Inklusion **echt**?

Mit Beispiel 7.1 ist

$$L := \{a^n b^n \mid n \geq 0\} \in \mathcal{L}_2.$$

Wir haben gezeigt (Beispiel 2.3), dass L nicht erkennbar/regulär ist; also folgt mit Satz 8.1: $L \notin \mathcal{L}_3$.



Regulär vs. kontextfrei

Beispiel 8.4

Weitere kontextfreie, nicht reguläre Sprache: $\{a^n b^m \mid n \neq m\}$

... wird erzeugt von der kontextfreien Grammatik $G = (N, \Sigma, P, S)$ mit

$$\begin{aligned} N = \{S, S_{\geq}, S_{\leq}\}, \quad \Sigma = \{a, b\}, \quad P = & \left\{ \begin{array}{ll} S \rightarrow aS_{\geq}, & S \rightarrow S_{\leq}b, \\ S_{\geq} \rightarrow aS_{\geq}b, & S_{\leq} \rightarrow aS_{\leq}b, \\ S_{\geq} \rightarrow aS_{\geq}, & S_{\leq} \rightarrow S_{\leq}b, \\ S_{\geq} \rightarrow \varepsilon, & S_{\leq} \rightarrow \varepsilon \end{array} \right\} \end{aligned}$$

Es gilt:

- $S_{\geq} \vdash_G^* w \in \{a, b\}^*$ $\Rightarrow w = a^n b^m$ mit $n \geq m$
- $S_{\leq} \vdash_G^* w \in \{a, b\}^*$ $\Rightarrow w = a^n b^m$ mit $n \leq m$

also:

- $S \vdash_G^* w \in \{a, b\}^*$ $\Rightarrow w = aa^n b^m$ ($n \geq m$) oder $w = a^n b^m b$ ($n \leq m$)

d.h. $L(G) = \{a^n b^m \mid n \neq m\}$



Teil I: Endliche Automaten und reguläre Sprachen

0. Grundbegriffe
1. Endliche Automaten
2. Nachweis der Nichterkennbarkeit
3. Abschlusseigenschaften
4. Entscheidungsprobleme
5. Reguläre Ausdrücke und Sprachen
6. Minimale DEAs und die Nerode-Rechtskongruenz

Teil II: Grammatiken, kontextfreie Sprachen und Kellerautomaten

- 
7. Die Chomsky-Hierarchie
 8. Rechtslineare Grammatiken und reguläre Sprachen
 9. Normalformen und Entscheidungsprobleme
 10. Abschlusseigenschaften und Pumping-Lemma
 11. Kellerautomaten
 12. Die Struktur kontextfreier Sprachen



§ 9 Normalformen und Entscheidungsprobleme für kontextfreie Grammatiken



Zur Erinnerung: Grammatiktypen

Typ 0: **allgemeine Grammatiken**

gehören in die Theorie der Berechenbarkeit (\vdash VL Berechenbark.)

Typ 1: **monotone Grammatiken**

auch Berechenbarkeit

Regeln nicht verkürzend,
mit Ausnahme für ε

Typ 2: **kontextfreie Grammatiken**

Regeln $A \rightarrow w$

Grundlage für die **Syntax von Programmiersprachen**,
Verwendung (in erweiterter Form) im **Compilerbau (Parser)**

Typ 3: **rechtslineare Grammatiken**

erzeugen genau die
regulären Sprachen

Regeln $A \rightarrow uB$

und $A \rightarrow u$, $u \in \Sigma^*$



§9.1 Vereinfachung kontextfreier Grammatiken und das Leerheitsproblem



Vereinfachung kontextfreier Grammatiken

Wir zeigen zunächst, wie man in einer kontextfreien Grammatik überflüssige Nichtterminalsymbole entfernen kann.

„Nebenbei“ sehen wir, wie man das Leerheitsproblem lösen kann.

Definition 9.1

Es sei $G = (N, \Sigma, P, S)$ eine kontextfreie Grammatik ([kfG](#)).

1. $A \in N$ heißt [terminierend](#), wenn es ein $w \in \Sigma^*$ gibt mit $A \vdash_G^* w$.
2. $A \in N$ heißt [erreichbar](#), wenn es $u, v \in (\Sigma \cup N)^*$ gibt mit $S \vdash_G^* uAv$.
3. G heißt [reduziert](#), wenn alle Elemente von N erreichbar und terminierend sind.

Zwei Grammatiken heißen [äquivalent](#), falls sie dieselbe Sprache erzeugen.



Terminierende Symbole

Lemma 9.2 (Berechnung der terminierenden Symbole)

Für jede kfG $G = (N, \Sigma, P, S)$ kann man
die Menge der terminierenden Nichtterminalsymbole berechnen.

Wir definieren dazu

$$T_1 := \{A \in N \mid \exists w \in \Sigma^* : A \longrightarrow w \in P\}$$

$$T_{i+1} := T_i \cup \{A \in N \mid \exists w \in (\Sigma \cup T_i)^* : A \longrightarrow w \in P\}$$

Es gilt: $T_1 \subseteq T_2 \subseteq T_3 \subseteq \dots \subseteq N$

Beispiel 9.3

$$\begin{aligned} P = \{ & S \longrightarrow A, \quad S \longrightarrow aCbBa, \quad A \longrightarrow aBAc, \quad B \longrightarrow Cab, \\ & C \longrightarrow AB, \quad C \longrightarrow aa \} \end{aligned}$$

$$T_1 = \{C\}$$

$$T_2 = \{C, B\}$$

$$T_3 = \{C, B, S\}$$

$$T_4 = \{C, B, S\}$$



Terminierende Symbole

Lemma 9.2 (Berechnung der terminierenden Symbole)

Für jede kfG $G = (N, \Sigma, P, S)$ kann man
die Menge der terminierenden Nichtterminalsymbole berechnen.

Wir definieren dazu

$$T_1 := \{A \in N \mid \exists w \in \Sigma^* : A \longrightarrow w \in P\}$$

$$T_{i+1} := T_i \cup \{A \in N \mid \exists w \in (\Sigma \cup T_i)^* : A \longrightarrow w \in P\}$$

Es gilt: $T_1 \subseteq T_2 \subseteq T_3 \subseteq \dots \subseteq N$

Da N endlich ist, gibt es ein k mit $T_k = T_{k+1}$ und es gilt: $T_k = \bigcup_{i \geq 1} T_i$

Behauptung: $T_k = \{A \in N \mid A \text{ ist terminierend}\}$



Terminierende Symbole

Zurück zu Beispiel 9.3:

$$P = \{ S \rightarrow A, \quad S \rightarrow aCbBa, \quad A \rightarrow aBAc, \quad B \rightarrow Cab, \\ C \rightarrow AB, \quad C \rightarrow aa \}$$

$$T_1 = \{C\} \quad T_2 = \{C, B\} \quad T_3 = \{C, B, S\} \quad T_4 = \{C, B, S\}$$

Es ist also **A** das einzige **nicht terminierende** Symbol.



Das Leerheitsproblem

Entscheidungsprobleme für kfG analog wie für Automaten, also z. B.:

Definition Leerheitsproblem

Gegeben: kfG G

Frage: Ist $L(G) = \emptyset$?

Satz 9.4

Das Leerheitsproblem für kfG ist **in polynomieller Zeit entscheidbar**.

Beweis. Offenbar gilt:

$L(G) \neq \emptyset$ gdw. $\exists w \in \Sigma^* : S \vdash_G^* w$ gdw. S ist terminierend

und Algo. zum Berechnen der termin. Symbole läuft in polynomieller Zeit

In VL Berechenbarkeit:

Leerheitsproblem für Typ-0/1-Grammatiken ist **nicht entscheidbar**.



Erreichbare Symbole

Lemma 9.5 (Berechnung der erreichbaren Symbole)

Für jede kfG $G = (N, \Sigma, P, S)$ kann man
die Menge der erreichbaren Nichtterminalsymbole berechnen.

Wir definieren dazu

$$E_0 := \{S\}$$

$$E_{i+1} := E_i \cup \{A \in N \mid \exists B \in E_i \text{ mit Regel } B \longrightarrow u_1 A u_2 \in P\}$$

Es gilt: $E_0 \subseteq E_1 \subseteq E_2 \subseteq \dots \subseteq N$

Da N endlich ist, gibt es ein k mit $E_k = E_{k+1}$ und es gilt: $E_k = \bigcup_{i \geq 0} E_i$

Behauptung: $E_k = \{A \in N \mid A \text{ ist erreichbar}\}$

„ \subseteq “ per Induktion über i : jedes E_i enthält nur erreichbare Symbole

„ \supseteq “ per Induktion über i : $S \vdash_G^{< i} uAv$ impliziert $A \in E_i$



Erreichbare Symbole

Beispiel 9.6

$$P = \{ S \rightarrow aS, \quad S \rightarrow SB, \quad S \rightarrow SS, \quad S \rightarrow \varepsilon, \\ A \rightarrow ASB, \quad A \rightarrow C, \quad B \rightarrow Cb \}$$

$$E_0 = \{S\} \quad E_1 = \{S, B\} \quad E_2 = \{S, B, C\} \quad E_3 = \{S, B, C\}$$

Es ist also **A** das einzige unerreichbare Symbol.



Lemma 9.7 (Konstruieren einer reduzierten kfG)

Zu jeder kfG G mit $L(G) \neq \emptyset$ kann man
eine äquivalente **reduzierte kfG** konstruieren.

Beachte:

Eine kfG G mit $L(G) = \emptyset$ kann **niemals** reduziert sein, da

- jede Grammatik ein Startsymbol S haben muss und
- S in G nicht terminierend sein kann.

Beweis.

Eliminiere **erst** nichtterminierende, **dann** unerreichbare Symbole.



Reduzierte kfG

Schritt 1. Entfernen nichtterminierender Symbole

Sei $G = (N, \Sigma, P, S)$.

$G' := (N', \Sigma, P', S)$ ist die Einschränkung von G auf terminierende Symbole:

$$N' := \{A \in N \mid A \text{ ist terminierend in } G\}$$

$$P' := \{A \rightarrow w \in P \mid w \in (N' \cup \Sigma)^*\}$$

Beachte: wegen $L(G) \neq \emptyset$ ist S terminierend, also $S \in N'$!

Schritt 2. Entfernen unerreichbarer Symbole

$G'' := (N'', \Sigma, P'', S)$ ist die Einschränkung von G' auf erreichbare Symbole:

$$N'' := \{A \in N' \mid A \text{ ist erreichbar in } G'\}$$

$$P'' := \{A \rightarrow w \in P' \mid A \in N''\}$$

Man sieht leicht: $L(G) = L(G') = L(G'')$ und G'' ist reduziert



Reihenfolge der beiden Schritte ist wichtig!

- Betrachte

$$S \rightarrow \varepsilon, \quad S \rightarrow AB, \quad A \rightarrow a$$

- Alle Symbole erreichbar;
zuerst Entfernen unerreichbarer Symbole ändert nichts.
- **danach** Entfernen nichtterminierender Symbole (**B**) liefert

$$S \rightarrow \varepsilon, \quad A \rightarrow a$$

- Diese Grammatik ist nicht reduziert, da nun **A nicht erreichbar**.



§9.2 Die Chomsky-Normalform



Normalformen

Es gibt verschiedene Normalformen für kfG:

Normalformen schränken die Form der erlaubten Regeln weiter ein; trotzdem können alle kontextfreien Sprachen erzeugt werden.

Wir betrachten Chomsky-Normalform (CNF):

Sie erlaubt nur Regeln der Form

$$A \rightarrow BC \quad \text{und} \quad A \rightarrow a$$

sowie $S \rightarrow \varepsilon$, wenn S nirgends rechts vorkommt.

Chomsky-Normalform dient uns

- als Basis für einen effizienten Algorithmus für das Wortproblem für kfG
- um $\mathcal{L}_2 \subseteq \mathcal{L}_1$ zu zeigen



Überführen einer kfG in Chomsky-Normalform

Wir gehen in drei Schritten vor:

- Entfernen **ε -Regeln** $A \rightarrow \varepsilon$ mit $A \neq S$
- Entfernen **Kettenregeln** $A \rightarrow B$
- Zerlegen Regeln $A \rightarrow w$ mit $w > 2$
und Aufheben der **Mischung** von Terminalen und Nichtterminalen



Entfernen der ε -Regeln

Definition 9.8 (ε -freie kfG)

Eine kfG heißt **ε -frei**, falls gilt:

1. Sie enthält keine **ε -Regeln** $A \longrightarrow \varepsilon$ mit $A \neq S$.
2. Enthält sie $S \longrightarrow \varepsilon$,
so kommt S nicht auf der rechten Seite einer Regel vor.



Entfernen der ε -Regeln

Lemma 9.9

Sei G eine kfG.

Dann lässt sich eine kfG G' ohne ε -Regeln konstruieren mit $L(G') = L(G) \setminus \{\varepsilon\}$.

Beweis.

(1) Finde alle $A \in N$ mit $A \vdash_G^* \varepsilon$:

$$N_1 := \{A \in N \mid A \longrightarrow \varepsilon \in P\}$$

$$N_{i+1} := N_i \cup \{A \in N \mid A \longrightarrow B_1 \cdots B_n \in P \text{ für } B_j \in N_i\}$$

Es gibt wieder ein k mit $N_k = N_{k+1}$.

Für dieses k gilt: $A \in N_k$ gdw. $A \vdash_G^* \varepsilon$ (Beweis wie gewohnt)



Entfernen der ε -Regeln

(2) Eliminiere in G alle Regeln $A \rightarrow \varepsilon$.

Um dies auszugleichen, **variiere alle übrigen Regeln**: jedes Vorkommen eines Nichtterminals $B \in N_k$ auf einer rechten Regelseite kann

- entweder **wegfallen** (weil $B \vdash_G^* \varepsilon$)
- oder **bleiben** (denn $B \vdash_G^* w$ für ein $w \neq \varepsilon$ ist ja auch möglich)

Präzise: für jede Regel

$$A \rightarrow u_1 B_1 \cdots u_n B_n u_{n+1}$$

mit $B_i \in N_k$ und $u_i \in (\Sigma \cup (N \setminus N_k))^*$, füge alle Regeln

$$A \rightarrow u_1 \beta_1 u_2 \cdots u_n \beta_n u_{n+1}$$

hinzu für beliebige $\beta_i \in \{B_i, \varepsilon\}$ mit $u_1 \beta_1 u_2 \cdots u_n \beta_n u_{n+1} \neq \varepsilon$.



Entfernen der ε -Regeln

Beispiel 9.10

$$P = \{ S \rightarrow aS, \quad S \rightarrow SS, \quad S \rightarrow bA, \quad A \rightarrow BB, \\ B \rightarrow CC, \quad B \rightarrow aAbC, \quad C \rightarrow \varepsilon \}$$

$$N_1 = \{C\} \quad N_2 = \{C, B\} \quad N_3 = \{C, B, A\} \quad N_4 = \{C, B, A\}$$

$$P' = \{ S \rightarrow aS, \quad S \rightarrow SS, \quad S \rightarrow bA, \quad \textcolor{red}{S \rightarrow b}, \\ \quad A \rightarrow BB, \quad \textcolor{red}{A \rightarrow B}, \\ \quad B \rightarrow CC, \quad \textcolor{red}{B \rightarrow C}, \quad B \rightarrow aAbC, \quad \textcolor{red}{B \rightarrow aAb}, \quad \textcolor{red}{B \rightarrow abC}, \quad \textcolor{red}{B \rightarrow ab} \\ \quad \cancel{\textcolor{red}{C \rightarrow \varepsilon}} \}$$

Die Ableitung $S \vdash_G bA \vdash_G bBB \vdash_G bCCB \vdash_G bCCCC \vdash_G^* b$
kann in G' direkt durch $\textcolor{red}{S \vdash_{G'} b}$ erreicht werden.



Entfernen der ε -Regeln

Satz 9.11

Zu jeder kfG G kann man eine äquivalente ε -freie kfG konstruieren.

Beweis.

- (1) Konstruiere G' wie im Beweis von Lemma 9.9 beschrieben.
- (2) Wenn $\varepsilon \notin L(G)$ (d. h. $S \not\vdash_G^* \varepsilon$, also $S \notin N_k$),
so ist G' die gesuchte ε -freie Grammatik.

Sonst erweitere G' um ein neues Startsymbol S_0 und die Produktionen

$$S_0 \rightarrow S \quad \text{und} \quad S_0 \rightarrow \varepsilon.$$



Entfernen der ε -Regeln

Korollar 9.12

$$\mathcal{L}_2 \subseteq \mathcal{L}_1$$

Beweis.

Offenbar ist jede ε -freie kfG eine Typ-1-Grammatik:

ε -freie kfG

Produktionen der Form

$$A \longrightarrow w$$

mit $|w| \geq 1$ oder

$$S \longrightarrow \varepsilon$$

und S nicht auf rechter Seite
einer Produktion

Typ-1-Grammatik

Produktionen der Form

$$u \longrightarrow w$$

mit $|w| \geq |u|$ oder

$$S \longrightarrow \varepsilon$$

und S nicht auf rechter Seite
einer Produktion



Entfernen der Kettenregeln

Satz 9.13

Zu jeder kfG G kann man eine äquivalente kfG konstruieren, die **keine Kettenregeln** enthält.

Beweis. Schritt 1. Bestimme Relation $K := \{(A, B) \in N \times N \mid A \vdash_G^* B\}$

$$K_0 := \{(A, A) \mid A \in N\}$$

$$K_{i+1} := K_i \cup \{(A, B) \in N \times N \mid \exists (A, B') \in K_i : B' \rightarrow B \in P\}$$

Es gibt ein k mit $K_k(A) = K_{k+1}(A)$ und für dieses k ist $K_k = K$.

Schritt 2. Entferne alle Kettenregeln

Um das auszugleichen, füge für jede verbleibende Regel $B \rightarrow w$ und jedes $(A, B) \in K$ auch $A \rightarrow w$ hinzu:

$$P' = \{A \rightarrow w \mid (A, B) \in K, B \rightarrow w \in P \text{ und } w \notin N\}$$



Entfernen der Kettenregeln

Beispiel 9.14

$$P = \{S \rightarrow A, A \rightarrow B, B \rightarrow aA, B \rightarrow b\}$$

$$K_0 = \{(S, S), (A, A), (B, B)\}$$

$$K_1 = \{(S, S), (A, A), (B, B), (S, A), (A, B)\}$$

$$K_2 = \{(S, S), (A, A), (B, B), (S, A), (A, B), (S, B)\}$$

$$K_3 = \{(S, S), (A, A), (B, B), (S, A), \underline{(A, B)}, \underline{(S, B)}\}$$

$$\begin{aligned} P' = & \{ B \rightarrow aA, \quad A \rightarrow aA, \quad S \rightarrow aA, \\ & B \rightarrow b, \quad \underline{A \rightarrow b}, \quad \underline{S \rightarrow b} \} \end{aligned}$$



Chomsky-Normalform

Satz 9.15 (Chomsky-Normalform)

Jede kfG lässt sich in eine äquivalente kfG in Chomsky-Normalform umformen, d. h. eine kfG, in der alle Regeln die folgende Form haben.

- $A \rightarrow a$ oder $A \rightarrow BC$ mit $A, B, C \in N$ und $a \in \Sigma$
- oder $S \rightarrow \varepsilon$, wobei dann S nirgends rechts vorkommt.

Beweis.

1. Konstruiere zur gegebenen kfG zunächst eine äquivalente ε -freie kfG und dann eine ohne Kettenregeln (Reihenfolge wichtig!)
2. Führe für jedes $a \in \Sigma$ ein neues Nichtterminalsymbol X_a und die Produktion $X_a \rightarrow a$ ein.



Chomsky-Normalform

Satz 9.15 (Chomsky-Normalform)

Jede kfG lässt sich in eine äquivalente kfG in Chomsky-Normalform umformen, d. h. eine kfG, in der alle Regeln die folgende Form haben.

- $A \rightarrow a$ oder $A \rightarrow BC$ mit $A, B, C \in N$ und $a \in \Sigma$
- oder $S \rightarrow \varepsilon$, wobei dann S nirgends rechts vorkommt.

Beweis.

3. Ersetze in jeder Produktion $A \rightarrow w$ mit $w \notin \Sigma$ alle Terminalsymbole a durch die zugehörigen X_a .

4. Produktionen $A \rightarrow B_1 \cdots B_n$ mit $n > 2$ werden ersetzt durch

$$A \rightarrow B_1 C_1, \quad C_1 \rightarrow B_2 C_2, \quad \dots, \quad C_{n-2} \rightarrow B_{n-1} B_n,$$

wobei die C_i jeweils neue Nichtterminale sind.



Chomsky-Normalform

Beispiel (für Schritte 2–4)

$$P = \{S \rightarrow aSbA, A \rightarrow ab, A \rightarrow b\}$$

Schritt 2

$$P = \{S \rightarrow aSbA, A \rightarrow ab, A \rightarrow b, X_a \rightarrow a, X_b \rightarrow b\}$$

Schritt 3

$$P = \{S \rightarrow X_a S X_b A, A \rightarrow X_a X_b, A \rightarrow b, X_a \rightarrow a, X_b \rightarrow b\}$$

Schritt 4

$$\begin{aligned} P = \{S \rightarrow X_a C_1, C_1 \rightarrow S C_2, C_2 \rightarrow X_b A, \\ A \rightarrow X_a X_b, A \rightarrow b, X_a \rightarrow a, X_b \rightarrow b\} \end{aligned}$$



Komplexitätsanalyse

Die präsentierte Umwandlung in Chomsky-NF kann **exponentiell aufblasen**

Ursache ist das Entfernen von ε -Regeln:

wenn $A \rightarrow B_1 \cdots B_n$ Regel und $B_1, \dots, B_n \in N_k$, dann

kann jedes B_i entweder bleiben oder eliminiert werden

$\rightsquigarrow 2^n - 1$ viele Regeln

Das lässt sich aber **leicht verhindern**:

- Schritt 4 **vor** den anderen beiden Schritten ausführen
- dann nur noch Regeln $A \rightarrow BC$ vom ε -Freimachen betroffen
daraus werden dann **höchstens 3** Regeln

Jede kfG lässt sich also in **polynomiell größere** kfG in Chomsky-NF wandeln

Die Umwandlung ist zudem in **polynomieller Zeit** möglich



§9.4 Die Greibach-Normalform (Skizze)



Greibach-Normalform

... ist eine weitere interessante Normalform.

Hier gibt es für jedes Wort in der Sprache eine Ableitung, die die Terminalsymbole streng **von links nach rechts** erzeugt (**genau ein Terminalsymbol pro Schritt!**).



Sheila Greibach
© University of Waterloo

Satz 9.16

Jede kfG lässt sich effektiv umformen in eine äquivalente kfG, die nur Regeln der folgenden Form enthält.

- $A \rightarrow aB_1 \cdots B_n \quad (A \in N, \ a \in \Sigma, \ B_1, \dots, B_n \in N, \ n \geq 0)$
- und eventuell $S \rightarrow \varepsilon$, wobei S nirgends rechts vorkommt

Eine derartige Grammatik ist in **Greibach-Normalform**.

(ohne Beweis)



Teil I: Endliche Automaten und reguläre Sprachen

0. Grundbegriffe
1. Endliche Automaten
2. Nachweis der Nichterkennbarkeit
3. Abschlusseigenschaften
4. Entscheidungsprobleme
5. Reguläre Ausdrücke und Sprachen
6. Minimale DEAs und die Nerode-Rechtskongruenz

Teil II: Grammatiken, kontextfreie Sprachen und Kellerautomaten

7. Die Chomsky-Hierarchie
8. Rechtslineare Grammatiken und reguläre Sprachen
9. Normalformen und → Entscheidungsprobleme
10. Abschlusseigenschaften und Pumping-Lemma
11. Kellerautomaten
12. Die Struktur kontextfreier Sprachen



§ 9.3 Das Wortproblem für kontextfreie Sprachen



Das Wortproblem für kontextfreie Sprachen

Definition

Sei G eine kfG. Das Wortproblem für $L = L(G)$ ist die Frage:

Gegeben: Wort w

Frage: Ist $w \in L(G)$?

Wir haben in §9.2 gesehen:

Jede kfG kann in äquivalente kfG in Chomsky-Normalform gewandelt werden

Zur Erinnerung: Chomsky-NF erlaubt nur Regeln der Form

$$A \longrightarrow BC \quad \text{und} \quad A \longrightarrow a$$

sowie $S \longrightarrow \varepsilon$, wenn S nirgends rechts vorkommt.

Für das Entscheiden des Wortproblems können wir also annehmen, dass G in Chomsky-Normalform ist.



Das Wortproblem für kontextfreie Sprachen

Für Grammatiken **in Chomsky-Normalform** gilt:

jede Ableitung eines Wortes $w \in \Sigma^*$ hat höchstens Länge $2|w|$:

- Produktionen der Form $A \rightarrow BC$ verlängern um 1,
werden also $(|w| - 1)$ -mal angewendet.
- Produktionen der Form $A \rightarrow a$ erzeugen genau ein Terminalsymbol,
werden also genau $|w|$ -mal angewendet.

Es gibt nur **endlich viele** Ableitungen der Länge $\leq 2|w|$.

Um das Wortproblem zu entscheiden, kann man im Prinzip

- alle diese Ableitungen aufzählen und
- prüfen, ob sie das gewünschte Wort w erzeugen.



Das Wortproblem für kontextfreie Sprachen

Das Aufzählverfahren hat im allgemeinen **exponentielle Laufzeit**, denn es kann $\geq 2^n$ Ableitungen der Länge n geben.

Thema dieses Abschnittes:

Polynomieller Algorithmus zum Entscheiden des Wortproblems für kfG in Chomsky-Normalform

Dieser Algorithmus ist bekannt als der **CYK-Algorithmus**

Seine Laufzeit ist n^3 mit n Länge des Eingabewortes

Beachte: Wir betrachten hier wieder **Datenkomplexität**,
d.h. die Grammatik wird als fest angenommen
(Vergl. Wortproblem für DEAs/NEAs)



Das Wortproblem für kontextfreie Sprachen

Definition 9.17

Sei $G = (N, \Sigma, P, S)$ eine kfG in Chomsky-NF und $w = a_1 \cdots a_n \in \Sigma^*$.

Für $i \leq j$ definieren wir $w_{ij} := a_i \cdots a_j$ und $N_{ij} := \{A \in N \mid A \vdash_G^* w_{ij}\}$.

Mit dieser Notation gilt nun:

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

2. $A \in N_{ii}$ gdw $A \vdash_G^* a_i$ gdw $A \rightarrow a_i \in P$

3. $A \in N_{ij}$ für $i < j$ gdw $A \vdash_G^* a_i \cdots a_j$

gdw es gibt $A \rightarrow BC \in P$ und $k \in \{i, \dots, j-1\}$

mit $B \vdash_G^* a_i \cdots a_k$ und $C \vdash_G^* a_{k+1} \cdots a_j$

gdw es gibt $A \rightarrow BC \in P$ und $k \in \{i, \dots, j-1\}$

mit $B \in N_{ik}$ und $C \in N_{(k+1)j}$

Dies liefert einen Divide-&-Conquer-Algorithmus zur Bestimmung von N_{1n}



Der CYK-Algorithmus



John Cocke
© NAE



Daniel Younger
© Univ. Waterloo



Tadao Kasami
© IEEE IT Society

Algorithmus (Cocke, Younger, Kasami)

```
for  $i := 1$  to  $n$  do          (Teilwörter der Länge 1)
   $N_{ii} := \{A \mid A \rightarrow a_i \in P\}$ 

for  $\ell := 1$  to  $n - 1$  do      (Teilwörter der Länge  $\ell + 1$ , also  $N_{ij}$  mit  $j - i = \ell$ )
  for  $i := 1$  to  $n - \ell$  do    (Startposition  $i$  von Teilwort  $w_{ij}$ )
     $j := i + \ell$               (Endposition  $j$  von Teilwort  $w_{ij}$ )
     $N_{ij} := \emptyset$ 
    for  $k := i$  to  $j - 1$  do  (mögliche Trennpositionen  $k$ )
       $N_{ij} := N_{ij} \cup \{A \mid A \rightarrow BC \in P \text{ mit } B \in N_{ik} \text{ und } C \in N_{(k+1)j}\}$ 
```

Beachte:

In der innersten Schleife sind N_{ik} und $N_{(k+1)j}$ bereits berechnet, denn $k - i$ und $j - (k + 1)$ sind kleiner als das aktuelle ℓ .



Der CYK-Algorithmus

Beispiel:

$$P = \{ S \rightarrow SA, \quad S \rightarrow a,$$
$$A \rightarrow BS,$$
$$B \rightarrow BB, \quad B \rightarrow BS, \quad B \rightarrow b, \quad B \rightarrow c \}$$
$$w = abacba$$


Lösen des Wortproblems mittels CYK-Algorithmus

Satz 9.18

Für jede kfG G in Chomsky-NF entscheidet der CYK-Algorithmus das Wortproblem für $L(G)$ in Zeit $O(|w|^3)$.

Beweis:

3 geschachtelte Schleifen, die jeweils $\leq |w| = n$ Iterationen durchlaufen
Daraus folgt: n^3 Schritte in der innersten Schleife □

Auch wenn G Teil der Eingabe ist, erhalten wir ein polynomielles Verfahren (egal ob G in Chomsky-NF oder nicht)

In VL Berechenbarkeit: Wortproblem ist

- nicht entscheidbar für Typ-0-Grammatiken
- entscheidbar aber (vermutlich) nicht Polyzeit für Typ-1-Grammatiken



Das Äquivalenzproblem für kfG

... ist analog wie für reguläre Sprachen/NEAs definiert:

Definition Äquivalenzproblem

Gegeben: kfG G_1, G_2

Frage: Ist $L(G_1) = L(G_2)$?

In der VL „Berechenbarkeit“ werden wir beweisen:

Satz

Das Äquivalenzproblem für kontextfreie Sprachen ist unentscheidbar.

Es gibt also keinen Algorithmus,

der für zwei gegebene kfG G_1 und G_2 entscheidet, ob $L(G_1) = L(G_2)$.



Zusammenfassung Entscheidungsprobleme

	Wortproblem	Leerheitsprob.	Äquivalenzprob.
Typ-0-Gramm.	unentscheidbar	unentscheidbar	unentscheidbar
Typ-1-Gramm.	entscheidbar, „nicht Polyzeit“	unentscheidbar	unentscheidbar
Typ-2-Gramm.	Polyzeit	Polyzeit	unentscheidbar
Typ-3-Gramm./ NEA/reg. Ausdr.	Linearzeit	Linearzeit	entscheidbar, „nicht Polyzeit“
DEA	Linearzeit	Linearzeit	Polyzeit



Teil I: Endliche Automaten und reguläre Sprachen

0. Grundbegriffe
1. Endliche Automaten
2. Nachweis der Nichterkennbarkeit
3. Abschlusseigenschaften
4. Entscheidungsprobleme
5. Reguläre Ausdrücke und Sprachen
6. Minimale DEAs und die Nerode-Rechtskongruenz

Teil II: Grammatiken, kontextfreie Sprachen und Kellerautomaten

7. Die Chomsky-Hierarchie
8. Rechtslineare Grammatiken und reguläre Sprachen
9. Normalformen und Entscheidungsprobleme
10. Abschlusseigenschaften und Pumping-Lemma
11. Kellerautomaten
12. Die Struktur kontextfreier Sprachen



§ 10 Abschlusseigenschaften und Pumping-Lemma



Abschlusseigenschaften

Satz 10.1

Die Klasse \mathcal{L}_2 der kontextfreien Sprachen ist unter
Vereinigung, **Konkatenation** und **Kleene-Stern** abgeschlossen.

Beweis.

Sei $L_1 = L(G_1)$ und $L_2 = L(G_2)$ für kfG $G_i = (N_i, \Sigma, P_i, S_i)$, $i = 1, 2$.

O. B. d. A. nehmen wir an, dass $N_1 \cap N_2 = \emptyset$; wählen neues $S \notin N_1 \cup N_2$.

1. Grammatik für $L_1 \cup L_2$:

$$G := (N_1 \cup N_2 \cup \{S\}, \Sigma, P_1 \cup P_2 \cup \{S \rightarrow S_1, S \rightarrow S_2\}, S)$$

2. Grammatik für $L_1 \cdot L_2$:

$$G' := (N_1 \cup N_2 \cup \{S\}, \Sigma, P_1 \cup P_2 \cup \{S \rightarrow S_1 S_2\}, S)$$

3. Grammatik für L_1^* :

$$G'' := (N_1 \cup \{S\}, \Sigma, P_1 \cup \{S \rightarrow \varepsilon, S \rightarrow S_1 S\}, S)$$

□



Abschlusseigenschaften

Wir werden zeigen:

Abschluss unter Schnitt und Komplement gilt **nicht!**

Wie zeigt man das?

Angabe von **Gegenbeispielen**:

z. B. kontextfreie Sprachen L_1, L_2 finden mit $L_1 \cap L_2$ **nicht kontextfrei**

Wie zeigt man, dass eine Sprache nicht kontextfrei ist?

Durch geeignete Variante des **Pumping Lemma**

Um es zu beweisen, verwenden wir **Ableitungsbäume**.



Ableitungsbäume

Beispiel 10.2

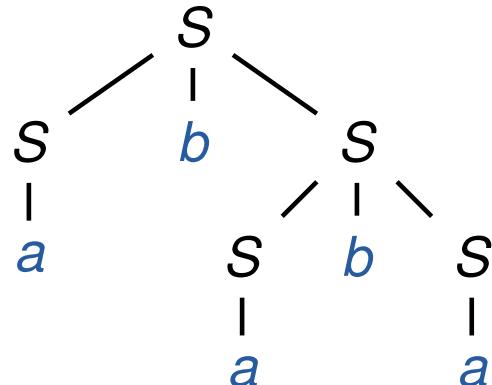
$$P = \{S \rightarrow SbS, S \rightarrow a\}$$

Drei Ableitungen von *ababa*:

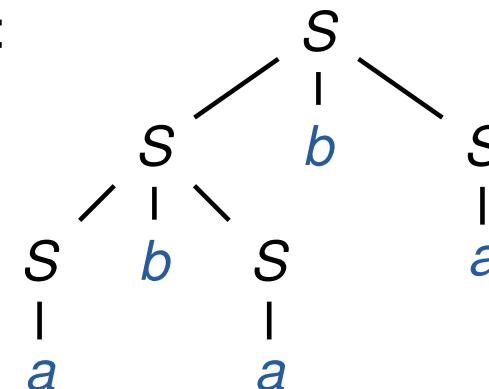
- (1) $S \vdash SbS \vdash abS \vdash abSbS \vdash ababS \vdash ababa$
- (2) $S \vdash SbS \vdash abS \vdash abSbS \vdash abSba \vdash ababa$
- (3) $S \vdash SbS \vdash Sba \vdash SbSba \vdash Sbaba \vdash ababa$

Die zugehörigen Ableitungsbäume:

(1), (2):



(3):



Ableitungsbäume

Ein Ableitungsbaum kann also für **mehrere Ableitungen** stehen; dasselbe Terminalwort kann **verschiedene Ableitungsbäume** haben.

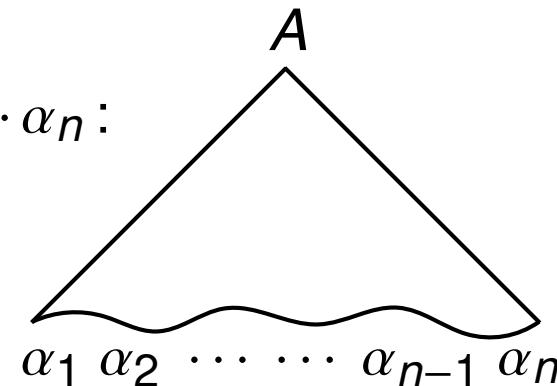
Allgemein:

Die **Knoten** des Ableitungsbaums sind mit Zeichen aus $\Sigma \cup N$ beschriftet.

Terminalsymbole dürfen als Beschriftung nur an den **Blättern** vorkommen, Nichtterminale **überall** (für partielle Ableitungen)

Ein mit A beschrifteter Knoten **kann** mit $\alpha_1, \dots, \alpha_n$ beschriftete **Nachfolgerknoten** haben, wenn $A \rightarrow \alpha_1 \cdots \alpha_n \in P$ ist.

Ableitungsbäum für Ableitung $A \vdash_G^* \alpha_1 \cdots \alpha_n$:



Pumping-Lemma

Lemma 10.3 (Pumping-Lemma für kontextfreie Sprachen)

Für jede kontextfreie Sprache L gibt es ein $n_0 \geq 1$, so dass gilt:
für jedes $z \in L$ mit $|z| \geq n_0$ existiert eine Zerlegung

$$z = uvwxy$$

mit:

- $vx \neq \varepsilon$ und $|vx| \leq n_0$
- $uv^iwx^i y \in L$ für alle $i \geq 0$

Beweis.

Sei L kontextfreie Sprache und G kfG mit $L(G) = L$.

O. B. d. A.: G ist ε -frei und ohne Kettenregeln.

Sei

- m die Anzahl der Nichtterminale in G
- k die maximale Länge von rechten Regelseiten in G

Wähle $n_0 = k^{m+1}$



Pumping-Lemma

Elementare Kombinatorik:

Jeder Baum der Tiefe $\leq t$ und Verzweigungsanzahl $\leq k$ hat $\leq k^t$ Blätter.

(1)

Sei $z \in L(G)$ mit $|z| \geq n_0 = k^{m+1}$. Betrachte Ableitungsbaum für z .

(2)

Der Baum hat $|z| \geq k^{m+1}$ Blätter, also gibt es Pfad der Länge $\geq m + 1$
(Pfad: Weg von Wurzel zu Blatt, Länge gemessen in Anzahl Kanten)

(3)

Auf diesem Pfad kommt mindestens ein Nichtterminal zweimal vor:
Pfad der Länge $\geq m + 1$ hat $> m + 1$ Knoten, davon $> m$ mit Nichtterminalen
Da es nur m Nichtterminale gibt, kommt mind. ein $A \in N$ zweimal vor.



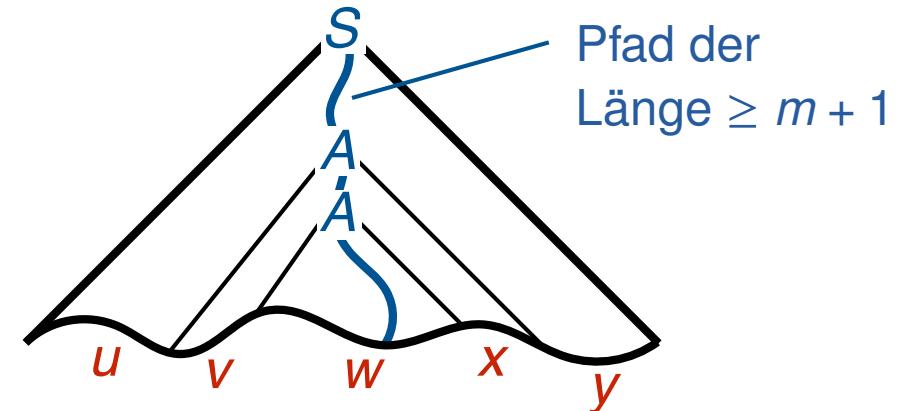
Pumping-Lemma

- (4) Wähle u, v, w, x, y wie folgt:

es gilt: $S \vdash_G^* uAy$

$A \vdash_G^* vAx$

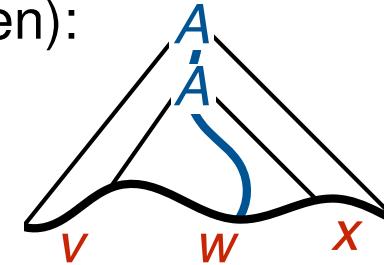
$A \vdash_G^* w$



woraus folgt: $S \vdash_G^* uAy \vdash_G^* uv^i Ax^j y \vdash_G^* uv^i wx^j y$ (Pumpen möglich!)

- (5) Wie gewünscht gilt $vx \neq \varepsilon$: sonst hätten wir $A \vdash_G^+ A$,
was wegen ε -Freiheit und ohne Kettenregeln nicht möglich ist.

- (6) Wählt man in (3) die **ersten beiden Vorkommen** eines Nicht-terminals A (von den **Blättern** aus gesehen):



so hat dieser Teilbaum Tiefe $\leq m + 1$, also $|vwx| \leq k^{m+1} = n_0$. □



Anwendung des Pumping-Lemmas

Lemma 10.4

$L := \{a^n b^n c^n \mid n \geq 1\}$ ist nicht kontextfrei.

Beweis.

Wir verwenden das Pumping-Lemma für kontextfreie Sprachen

Angenommen, L wäre doch kontextfrei

Dann gibt es ein n_0 wie im Pumping Lemma

Betrachte das Wort $z := a^{n_0} b^{n_0} c^{n_0} \in L$ (offensichtlich ist $|z| \geq n_0$).



Anwendung des Pumping-Lemmas

Sei $z = uvwxy$ Zerlegung von $z = a^{n_0} b^{n_0} c^{n_0}$ mit $vx \neq \epsilon$ und $|vwx| \leq n_0$.

1. Fall:

v enthält verschiedene Symbole aus $\Sigma = \{a, b, c\}$

wähle $i = 2 \rightsquigarrow$ dann $uv^2wx^2y \notin a^*b^*c^* \supseteq L \downarrow$

2. Fall:

x enthält verschiedene Symbole

\rightsquigarrow führt zu analogem Widerspruch \downarrow

3. Fall:

v enthält lauter gleiche Symbole, x ebenso

\rightsquigarrow Dann kommt ein Symbol aus $\{a, b, c\}$ nicht in vx vor, sagen wir a .

wähle $i = 0$

$\rightsquigarrow uv^0wx^0y$ enthält n_0 mal a , ist aber kürzer als $3n_0$, also nicht in L . $\downarrow \square$



Nützliche Konsequenz des Pumping-Lemmas

Satz 10.5
 $\mathcal{L}_2 \subset \mathcal{L}_1$

d.h.: Klasse der kontextfreien Sprachen ist **echte Teilmenge**
der Klasse der kontextsensitiven Sprachen

Beweis.

Wir haben bereits gezeigt: $\mathcal{L}_2 \subseteq \mathcal{L}_1$ (Korollar 9.12, ε -Freiheit)

Echtheit der Inklusion folgt aus:

$$L = \{a^n b^n c^n \mid n \geq 1\} \in \mathcal{L}_1 \setminus \mathcal{L}_2$$

$L \notin \mathcal{L}_2$: Lemma 10.4

$L \in \mathcal{L}_1$: Beispiel 7.3 (Typ 1 Grammatik G mit $L(G) = L$)

$G = (N, \Sigma, P, S)$ mit $N = \{S, B\}$, $\Sigma = \{a, b, c\}$ und

$$P = \{S \rightarrow aSBc, S \rightarrow abc, cB \rightarrow Bc, bB \rightarrow bb\}$$



Weitere Konsequenz: Abschlusseigenschaften

Korollar 10.6

\mathcal{L}_2 (Klasse der kontextfreien Sprachen)

ist **nicht** unter **Schnitt** und **Komplement** abgeschlossen.

Beweis.

(1) Die Sprachen

$$\{a^n b^n c^m \mid n \geq 1, m \geq 1\} = \{a^n b^n \mid n \geq 1\} \cdot \{c\}^+$$

$$\{a^m b^n c^n \mid n \geq 1, m \geq 1\} = \{a\}^+ \cdot \{b^n c^n \mid n \geq 1\}$$

sind **kontextfrei**:

- $\{a^n b^n \mid n \geq 1\} \in \mathcal{L}_2$ und $\{c\}^+ \in \mathcal{L}_3 \subseteq \mathcal{L}_2$
- $\{a\}^+ \in \mathcal{L}_3 \subseteq \mathcal{L}_2$ und $\{b^n c^n \mid n \geq 1\} \in \mathcal{L}_2$
- \mathcal{L}_2 ist unter Konkatenation abgeschlossen.



Weitere Konsequenz: Abschlusseigenschaften

(2) $\{a^n b^n c^m \mid n, m \geq 1\} \cap \{a^m b^n c^n \mid n, m \geq 1\} = \{a^n b^n c^n \mid n \geq 1\}$

Wäre \mathcal{L}_2 unter \cap abgeschlossen, dann wäre $\{a^n b^n c^n \mid n \geq 1\} \in \mathcal{L}_2$. 

(3) $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$

Wäre \mathcal{L}_2 unter Komplement abgeschlossen, so auch unter \cap .  



Zusammenfassung Abschlusseigenschaften

\cap

\cup

-

•

*

Typ 0

kontext-
sensitiv

kontext-
frei

✗

✓

✗

✓

✓

regulär

✓

✓

✓

✓

✓



Teil I: Endliche Automaten und reguläre Sprachen

0. Grundbegriffe
1. Endliche Automaten
2. Nachweis der Nichterkennbarkeit
3. Abschlusseigenschaften
4. Entscheidungsprobleme
5. Reguläre Ausdrücke und Sprachen
6. Minimale DEAs und die Nerode-Rechtskongruenz

Teil II: Grammatiken, kontextfreie Sprachen und Kellerautomaten

7. Die Chomsky-Hierarchie
8. Rechtslineare Grammatiken und reguläre Sprachen
9. Normalformen und Entscheidungsprobleme
10. Abschlusseigenschaften und Pumping-Lemma
11. Kellerautomaten
12. Die Struktur kontextfreier Sprachen

