Grundlagen der Theoretischen Informatik

4. Kellerautomaten und kontextfreie Sprachen (III)

24.05.2017

Viorica Sofronie-Stokkermans

e-mail: sofronie@uni-koblenz.de

Übersicht

1. Motivation
2. Terminologie
3. Endliche Automaten und reguläre Sprachen
4. Kellerautomaten und kontextfreie Sprachen
5. Turingmaschinen und rekursiv aufzählbare Sprachen
6. Berechenbarkeit, (Un-)Entscheidbarkeit
7. Komplexitätsklassen P und NP

Umformung von Grammatiken

- **Startsymbol nur links**

  Ist das bei einer Grammatik nicht gegeben, kann man es wie folgt erreichen:

  – Führe ein neues Startsymbol S_{neu} ein

  – Füge die Regel $S_{neu} \to S$ hinzu.

- **Keine nutzlose Symbole**

Theorem (cf-Grammatik ohne nutzlose Symbole)

Ist $G = (V, T, R, S)$ eine cf-Grammatik mit $L(G) \neq \emptyset$,
dann existiert eine cf-Grammatik $G' = (V', T', R', S')$ mit:

- G' ist äquivalent zu G.
- Jedes $x \in (V \cup T)$ ist erreichbar und co-erreichbar.

Normalform für Regeln

Theorem (Normalform)
Zu jeder Grammatik G (beliebigen Typs) existiert eine äquivalente Grammatik G', bei der für alle Regeln $P \to Q \in R'$ gilt:

- $Q \in V^*$ und P beliebig
- $Q \in T$ und $P \in V$

Für alle Typen außer den linearen hat G' denselben Typ wie G.

Beweis: Für jedes $t \in T$ erzeuge man eine neue Variable V_t.

- $V' = V \cup \{V_t \mid t \in T\}$
- R' entsteht aus R, indem für jede Regel $P \to Q \in R$ in Q alle Vorkommen eines Terminals t durch die zugehörige Variable V_t ersetzt werden. Außerdem enthält R' für jedes $t \in T$ eine neue Regel $V_t \to t$.

Elimination von ε-Regeln

Definition (ε-Regel, nullbare Variablen)
Eine Regel der Form $P \to \varepsilon$ (P eine Variable) heißt **ε-Regel**.

Eine Variable A heißt **nullbar**, falls $A \Longrightarrow^* \varepsilon$

Theorem (ε-Regeln sind eliminierbar)
Zu jeder cf-Grammatik G existiert eine äquivalente cf-Grammatik G'

- ohne ε-Regeln und nullbare Variablen,
  falls $\varepsilon \notin L(G)$,
- mit der einzigen ε-Regel $S \to \varepsilon$ und der einzigen nullbaren Variablen S,
  falls $\varepsilon \in L(G)$ und S das Startsymbol ist.

Elimination von ε-Regeln

Beweis (Forts.)

Ausgangsgrammatik G habe die Normalform, bei der für jede Regel $P \to Q$: $Q \in V^*$ oder $Q \in T$.

Für jede Regel $P \to A_1 \dots A_n$ generiere alle möglichen Kombinationen

$$P \to \alpha_1 \dots \alpha_n$$

mit

- $\alpha_i \in \{\varepsilon, A_i\}$ falls A_i nullbar
- $\alpha_i = A_i$ falls A_i nicht nullbar

Dann

- Füge alle diese neuen Regeln zur Grammatik hinzu
- Entferne alle Regeln der Form $A \to \varepsilon$ mit $A \neq S$

Elimination von ε-Regeln

Beweis ((Forts.)

Zu zeigen:
Für die neue Grammatik G' gilt: $L(G') = L(G)$

Vorgehen:

- G hat die Normalform:

  Für jede Regel $P \to Q$ gilt $Q \in V^*$ oder $Q \in T$.

- Wir beweisen die etwas stärkere Behauptung

$$\text{für alle } A \in V \;\; \text{für alle } w \in (V \cup T)^* - \{\varepsilon\}$$
$$\big((A \Longrightarrow_G^* w) \quad \underline{\text{gdw}} \quad (A \Longrightarrow_{G'}^* w)\big),$$

- Daraus folgt sofort $L(G') = L(G)$.

Elimination von ε-Regeln

Beweis (Forts.)

"$\Rightarrow$" Wir zeigen: Aus $A \Longrightarrow_G^* w$ folgt $A \Longrightarrow_{G'}^* w$ (Induktion über Länge einer Ableitung von A nach w in G).

Induktionsanfang: Länge $= 0$.

Dann ist $w = A$, und $A \Longrightarrow_{G'}^* A$ gilt immer.

Induktionsschritt: Es sei schon gezeigt: Wenn in G in n Schritten eine Ableitung $B \Longrightarrow_G^* u$ durchgeführt werden kann, dann folgt, dass in G' die Ableitung $B \Longrightarrow_{G'}^* u$ möglich ist.

Elimination von ε-Regeln

Beweis (Forts.)

Außerdem gelte in der Ausgangsgrammatik G: $A \Longrightarrow_G^* w \neq \varepsilon$ in $n+1$ Schritten.

Dann gilt:

- $A \Longrightarrow_G w' \Longrightarrow_G^* w$,
- $w' = A_1 \ldots A_\ell \Longrightarrow_G^* w_1 \ldots w_\ell = w$,
- und es wird jeweils A_i zu w_i in höchstens n Schritten für geeignete $w', A_1, \ldots, A_\ell, w_1, \ldots, w_\ell$.
- Für $1 \leq i \leq \ell$ gilt:
  - Entweder $w_i \neq \varepsilon$ und $A_i \Longrightarrow_G^* w_i$
    also (per Induktionsvoraussetzung) $A_i \Longrightarrow_{G'}^* w_i$
  - oder $w_i = \varepsilon$ und $A_i \Longrightarrow_G^* w_i$.

Elimination von ε-Regeln

Beweis (Forts.)

Fall 1: $w_i = \varepsilon$, A_i ist nullbar.

Dann gibt es in G' eine Regel $A \to A_1 \dots A_{i-1} A_{i+1} \dots A_\ell$ nach der obigen Konstruktionsvorschrift für G', falls $A_1 \dots A_{i-1} A_{i+1} \dots A_\ell \neq \varepsilon$. Das ist der Fall, denn sonst hätten wir: $A \Longrightarrow w' = \varepsilon \Longrightarrow^* w = \varepsilon$ (aus nichts wird nichts), aber $w = \varepsilon$ ist ausgeschlossen.

Fall 2: $w_i \neq \varepsilon$. Dann gilt nach Induktionsvoraussetzung

$A_i \Longrightarrow^*_{G'} w_i$.

Elimination von ε-Regeln

Beweis (Forts.)

Wir haben also folgendes gezeigt:

Sei $I = \{i \in \{1 \ldots \ell\} \mid w_i \neq \varepsilon\} \neq \emptyset$.

Dann gibt es in R' eine Regel $A \to A_{i_1} \ldots A_{i_m}$ mit $I = \{i_1, \ldots, i_m\}$, und die A_i sind so angeordnet wie in der ursprünglichen Regel $A \to A_1 \ldots A_\ell$.

Mit dieser neuen Regel können wir w so ableiten:

$$A \Longrightarrow_{G'} A_{i_1} \ldots A_{i_m} \Longrightarrow^*_{G'} w_{i_1} \ldots w_{i_m} = w$$

Elimination von ε-Regeln

Beweis (Forts.)

"$\Leftarrow$" Wir zeigen: Aus $A \Longrightarrow^*_{G'} w$ folgt $A \Longrightarrow^*_G w$ (Induktion über Länge einer Ableitung von A nach w in G'):

Induktionsanfang: Länge $= 0$. Dann ist $w = A$, und $A \Longrightarrow^*_G A$ gilt immer.

Induktionsschritt: Es gelte für alle Ableitungen $A \Longrightarrow^*_{G'} w$ einer Länge von höchstens n, dass $A \Longrightarrow^*_G w$.

Ist $A \Longrightarrow^*_{G'} w$ eine Ableitung der Länge $n+1$, so gibt es ein ℓ, Wörter $w_1, \ldots, w_\ell$ und Variablen $A_1, \ldots, A_\ell$ mit $A \Longrightarrow_{G'} A_1 \ldots A_\ell \Longrightarrow^*_{G'} w = w_1 \ldots w_\ell$. Es gilt jeweils $A_i \Longrightarrow^*_{G'} w_i$ in höchstens n Schritten, und $w_i \neq \varepsilon$.

Elimination von ε-Regeln

Beweis (Forts.)

Nach der Induktionsvoraussetzung folgt daraus:

- für die Originalgrammatik G gibt es Ableitungen $A_i \Longrightarrow_G^* w_i$
- damit gibt es auch eine Ableitung $A_1 \ldots A_\ell \Longrightarrow_G^* w$.

Elimination von ε-Regeln

Beweis (Forts.)

Nach der Induktionsvoraussetzung folgt daraus:

- für die Originalgrammatik G gibt es Ableitungen $A_i \Longrightarrow_G^* w_i$
- damit gibt es auch eine Ableitung $A_1 \ldots A_\ell \Longrightarrow_G^* w$.

Da es in G' eine Ableitung $A \Longrightarrow_{G'} A_1 \ldots A_\ell$ gibt, gibt es in R' eine Regel $A \to A_1 \ldots A_\ell$. Wie ist diese Regel aus R entstanden?

Elimination von ε-Regeln

Beweis (Forts.)

Nach der Induktionsvoraussetzung folgt daraus:

- für die Originalgrammatik G gibt es Ableitungen $A_i \Longrightarrow_G^* w_i$
- damit gibt es auch eine Ableitung $A_1 \dots A_\ell \Longrightarrow_G^* w$.

Da es in G' eine Ableitung $A \Longrightarrow_{G'} A_1 \dots A_\ell$ gibt, gibt es in R' eine Regel $A \to A_1 \dots A_\ell$. Wie ist diese Regel aus R entstanden?

Eine Regel in R' entsteht aus einer Regel in R, indem einige nullbare Variablen gestrichen werden. Es gab also in G nullbare Variablen B_1 bis B_m, so dass R die Regel

$$A \to A_1 \dots A_{\ell_1} B_1 A_{\ell_1+1} \dots A_{\ell_2} B_2 \dots A_m B_m A_{m+1} \dots A_\ell$$

enthält. (m kann auch 0 sein, dann war die Regel selbst schon in R.)

Elimination von ε-Regeln

Beweis (Forts.)

Also gilt in G:

$$\begin{aligned} A &\Longrightarrow_G A_1 \dots A_{\ell_1} B_1 A_{\ell_1+1} \dots A_{\ell_2} B_2 \dots A_m B_m A_{m+1} \dots A_\ell \\ &\Longrightarrow_G^* A_1 \dots A_{\ell_1} A_{\ell_1+1} \dots A_{\ell_2} \dots A_m A_{m+1} \dots A_\ell \Longrightarrow_G^* w \end{aligned}$$

da ja $B_i \Longrightarrow_G^* \varepsilon$ möglich ist. $\square$

Elimination von ε-Regeln: Beispiel

R :	R' :
$S \rightarrow ABD$	$S \rightarrow ABD \mid AD \mid BD \mid D$
$A \rightarrow ED \mid BB$	$A \rightarrow ED \mid BB \mid B$
$B \rightarrow AC \mid \varepsilon$	$B \rightarrow AC \mid A \mid C$
$C \rightarrow \varepsilon$	
$D \rightarrow d$	$D \rightarrow d$
$E \rightarrow e$	$E \rightarrow e$

Elimination von ε-Regeln: Beispiel

$$
\begin{array}{lcl}
 & R: & \\
S & \rightarrow & ABD \\
A & \rightarrow & ED \mid BB \\
B & \rightarrow & AC \mid \varepsilon \\
C & \rightarrow & \varepsilon \\
D & \rightarrow & d \\
E & \rightarrow & e
\end{array}
\qquad\qquad
\begin{array}{lcl}
 & R': & \\
S & \rightarrow & ABD \mid AD \mid BD \mid D \\
A & \rightarrow & ED \mid BB \mid B \\
B & \rightarrow & AC \mid A \mid C \\
 & & \\
D & \rightarrow & d \\
E & \rightarrow & e
\end{array}
$$

Für die Regelmenge R in der linken Spalte sind die Variablen A, B, C nullbar.

Elimination von ε-Regeln: Beispiel

$$\begin{array}{rcl} & R: & \\ S & \to & ABD \\ A & \to & ED \mid BB \\ B & \to & AC \mid \varepsilon \\ C & \to & \varepsilon \\ D & \to & d \\ E & \to & e \end{array} \qquad\qquad \begin{array}{rcl} & R': & \\ S & \to & ABD \mid AD \mid BD \mid D \\ A & \to & ED \mid BB \mid B \\ B & \to & AC \mid A \mid C \\ & & \\ D & \to & d \\ E & \to & e \end{array}$$

Für die Regelmenge R in der linken Spalte sind die Variablen A, B, C nullbar.

Der obige Algorithmus erzeugt aus R die rechts aufgeführte Regelmenge R'.

Elimination von ε-Regeln

Beobachtung

- Der Algorithmus lässt nutzlose Variablen zurück,
  die nicht in Prämissen auftauchen
  (und deshalb nicht co-erreichbar sind).
  Hier: C.
- Der Algorithmus lässt nutzlose Regeln zurück.
  Hier: $B \to AC \mid C$.

Elimination von ε-Regeln

Korollar.

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

Das heißt, jede kontextfreie Sprache ist auch kontextsensitiv

Elimination von ε-Regeln

Korollar.

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

Das heißt, jede kontextfreie Sprache ist auch kontextsensitiv

Beweis. Regeln einer kontextsensitiven Grammatik müssen folgende Form haben:

- entweder $uAv \to u\alpha v$
  mit $u, v, \alpha \in (V \cup T)^*, |\alpha| \geq 1, A \in V$
- oder $S \to \varepsilon$
  und S kommt in keiner Regelconclusio vor.

Diesen Bedingungen genügt die kontextfreie Grammatik nach Elimination der ε-Regeln.

Elimination von Kettenproduktionen

Definition. Eine Regel der Form

$$A \to B \qquad \text{mit } A, B \in V$$

heißt **Kettenproduktion**.

Theorem. (Kettenproduktionen sind eliminierbar)
Zu jeder cf-Grammatik existiert eine äquivalente cf-Grammatik ohne Kettenproduktionen.

Elimination von Kettenproduktionen

Beweis.

Sei $G = (V, T, R, S)$ eine kontextfreie Grammatik
ohne ε-Regeln, außer ggf. $S \to \varepsilon$.

Konstruiere neue Grammatik wie folgt:

1. Für alle
   - Variablenpaare $A, B \in V, \quad A \neq B \quad$ mit $A \Longrightarrow^* B$
   - Regeln $B \to \alpha \in R, \quad \alpha \notin V$

   füge zu R hinzu:

   $$A \to \alpha$$

2. Lösche alle Kettenproduktionen

Normalform für cf-Grammatiken

Theorem. Zu jeder cf-Grammatik existiert eine äquivalente cf-Grammatik

- ohne ε-Regeln
  (bis auf $S \to \varepsilon$, falls ε zur Sprache gehört;
  in diesem Fall darf S in keiner Regelconclusio vorkommen),
- ohne nutzlose Symbole,
- ohne Kettenproduktionen,
- so dass für jede Regel $P \to Q$ gilt: entweder $Q \in V^*$ oder $Q \in T$.

Normalform für cf-Grammatiken

Beweis.

1. Man teste zunächst, ob S nullbar ist. Falls ja, dann verwende man S_{neu} als neues Startsymbol und füge die Regeln $S_{neu} \to S \mid \varepsilon$ zum Regelsatz hinzu.
2. Man eliminiere nutzlose Symbole.
3. Man eliminiere alle ε-Regeln außer $S_{neu} \to \varepsilon$.
4. Man bringe die Grammatik in die Normalform,
   bei der für jede Regel $P \to Q$ gilt: entweder $Q \in V^*$ oder $Q \in T$.
5. Man eliminiere Kettenproduktionen.
6. Zum Schluss eliminiere man noch einmal alle nutzlosen Symbole
   (wg. Schritt 3)

Normalformen

Unterschied: Grammatiktypen und Normalformen

Gemeinsamkeit: Sowohl Grammatiktypen als auch Normalformen schränken die Form von Grammatikregeln ein.

Unterschied:

- Grammatiktypen (rechtslinear, kontextfrei usw.) führen zu **unterschiedlichen Sprachklassen**
- Normalformen führen zu **denselben Sprachklassen**

Normalformen

Wozu dann Normalformen?

- Weniger Fallunterscheidungen bei Algorithmen,
  die mit Grammatiken arbeiten.

- Struktur von Grammatiken einfacher zu “durchschauen”

Normalformen

Wozu dann Normalformen?

- Weniger Fallunterscheidungen bei Algorithmen, die mit Grammatiken arbeiten.
- Struktur von Grammatiken einfacher zu „durchschauen“

Zwei Normalformen

Chomsky-Normalform: Baut auf den Umformungen des vorigen Teils auf.

Greibach-Normalform: Ähnlich den rechtslinearen Grammatiken.

Chomsky-Normalform

Definition. Eine cf-Grammatik $G = (V, T, R, S)$ ist in **Chomsky-Normalform (CNF)**, wenn gilt:

- G hat nur Regeln der Form

$$\begin{array}{lll} A \to BC & \text{mit } A, B, C \in V \text{ und} & \\ A \to a & \text{mit } A \in V,\ a \in T & (\text{nicht } \varepsilon!) \end{array}$$

- Ist $\varepsilon \in L(G)$, so darf G zusätzlich die Regel $S \to \varepsilon$ enthalten. In diesem Fall darf S in keiner Regelconclusio vorkommen.
- G enthält keine nutzlosen Symbole.

Chomsky-Normalform

Theorem. Zu jeder cf-Grammatik existiert eine äquivalente cf-Grammatik in Chomsky-Normalform.

Chomsky-Normalform

Theorem. Zu jeder cf-Grammatik existiert eine äquivalente cf-Grammatik in Chomsky-Normalform.

Beweis.

Schritt 1: Wende auf G die Umformungen des letzten Abschnitts an.

Ergebnis:

- G hat keine nutzlosen Symbole
- Alle Regeln haben die Form
  1. $A \to \alpha$ mit $A \in V$ und $\alpha \in V^*$, $|\alpha| \geq 2$, und
  2. $A \to a$ mit $A \in V$, $a \in T$

Chomsky-Normalform

Beweis (Forts.)

Schritt 2: Regeln so umformen, dass keine Conclusio eine Länge größer 2 hat.
Ersetze jede Regel

$$A \to A_1 \dots A_n \text{ mit } A, A_i \in V, n \geq 3$$

durch:

$$\begin{array}{lcl} A & \to & A_1 C_1 \\ C_1 & \to & A_2 C_2 \\ & \vdots & \\ C_{n-2} & \to & A_{n-1} A_n \end{array}$$

Dabei sind die C_i neue Variablen in V.

$\square$

Greibach-Normalform

Definition.
Eine cf-Grammatik $G = (V, T, R, S)$ ist in **Greibach-Normalform (GNF)**, wenn gilt:

- G hat nur Regeln der Form

  $$A \to a\alpha \text{ mit } A \in V \text{ und } a \in T \text{ und } \alpha \in V^*$$

- Ist $\varepsilon \in L(G)$, so darf G zusätzlich die Regel $S \to \varepsilon$ enthalten.
  In diesem Fall darf S in keiner Regelconclusio vorkommen.
- G enthält keine nutzlosen Symbole.

Pumping-Lemma für kontextfreie Sprachen

Wiederholung

Theorem (Pumping-Lemma für L_3-Sprachen)
Sei $L \in \mathbf{RAT}$.
Dann existiert ein $n \in \mathbb{N}$, so dass:

Für alle

$$x \in L \quad \text{mit} \quad |x| \geq n$$

existiert eine Zerlegung

$$x = uvw \qquad u, v, w \in \Sigma^*$$

mit

- $|v| \geq 1$
- $|v| < n$
- $uv^m w \in L \quad$ für alle $m \in \mathbb{N}$

Pumping-Lemma für kontextfreie Sprachen

Theorem (Pumping-Lemma für kontextfreie Sprachen)
Sei L kontextfrei.
Dann existiert ein $n \in \mathbb{N}$, so dass:

Für alle

$$z \in L \quad \text{mit} \quad |z| \geq n$$

existiert eine Zerlegung

$$z = uvwxy \qquad u, v, w, x, y \in \Sigma^*$$

mit

- $|vx| \geq 1$
- $|vwx| < n$
- $uv^m wx^m y \in L$ für alle $m \in \mathbb{N}$

Pumping-Lemma für kontextfreie Sprachen

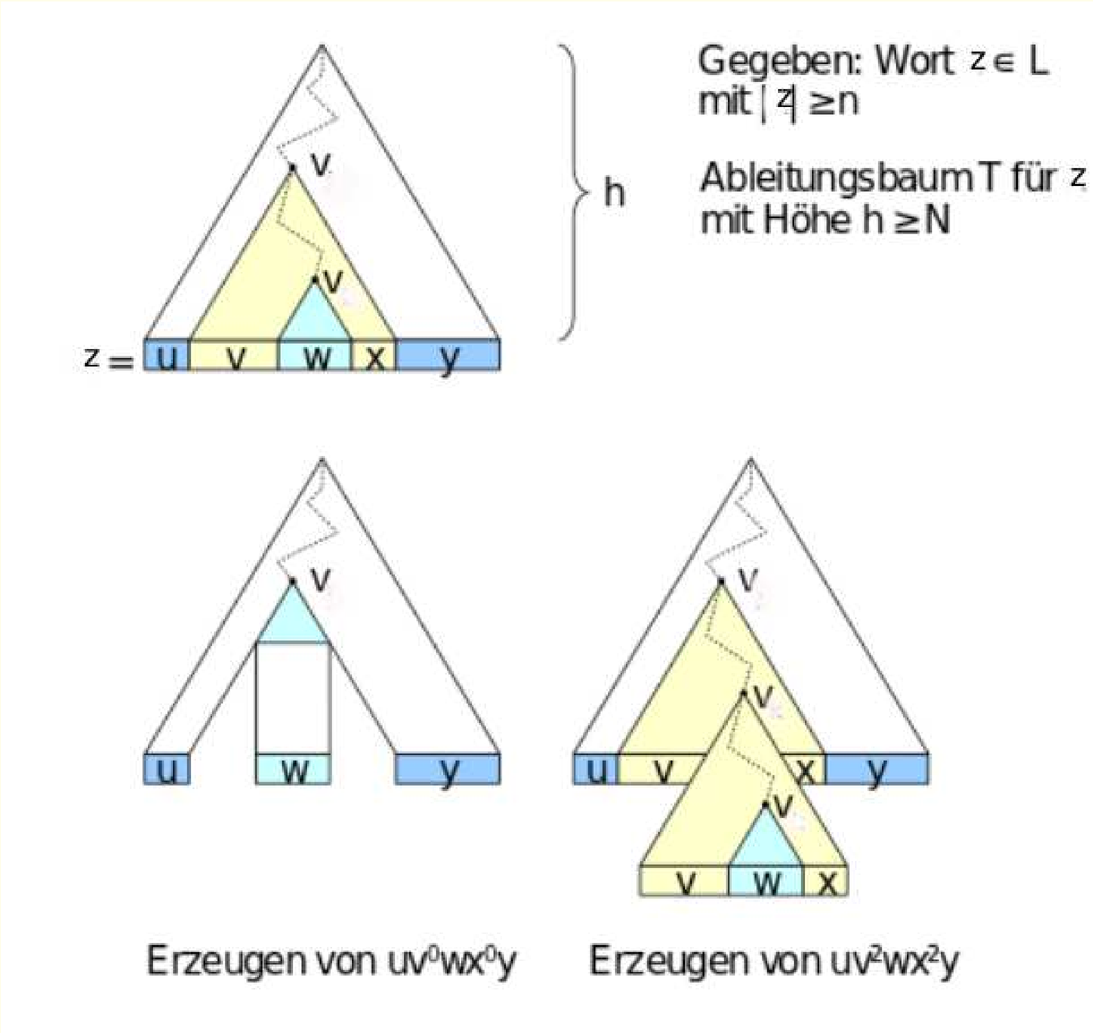
Beweisidee:

Bei der Ableitung eines hinreichend langen Wortes muss es eine Variable geben, die mehr als einmal auftaucht.

Dies führt zu einer Schleife in der Ableitung, die aufgepumpt werden kann.

Pumping-Lemma für kontextfreie Sprachen

Beweisidee:



Pumping-Lemma für kontextfreie Sprachen

Anwendung des Pumping-Lemmas für cf-Sprachen

Wenn das cf-Pumping-Lemma für eine Sprache nicht gilt,
dann kann sie nicht kontextfrei sein.

Pumping-Lemma für kontextfreie Sprachen

Anwendung des Pumping-Lemmas für cf-Sprachen

Wenn das cf-Pumping-Lemma für eine Sprache nicht gilt,
dann kann sie nicht kontextfrei sein.

Beispiel (Sprachen, die nicht kontextfrei sind)

Für folgende Sprachen kann man mit Hilfe des cf-Pumping-Lemmas zeigen, dass sie nicht kontextfrei sind:

- $\{a^p \mid p \text{ prim}\}$
- $\{a^n b^n c^n \mid n \in \mathbb{N}\}$
- $\{zzz \mid z \in \{a, b\}^*\}$.

Pumping-Lemma für kontextfreie Sprachen

$L_1 = \{a^p \mid p \text{ prim}\}$ ist nicht kontextfrei.

Beweis: Wir nehmen an, L_1 sei kontextfrei.

Sei dann n die zugehörige Konstante aus dem Pumping-Lemma.

Wir betrachten das Wort $z = a^p$, wobei p prim und $p \geq n + 2$.

Es muss dann eine Zerlegung $z = uvwxy$ geben, so dass:
$|vx| \geq 1, \quad |vwx| < n, \quad uv^i wx^i y \in L_1$ für alle $i \geq 0$.

Dann $u = a^{i_1}, v = a^{i_2}, w = a^{i_3}, x = a^{i_4}, y = a^{i_5}$ mit

- $i_1 + i_2 + i_3 + i_4 + i_5 = p$
- $i_2 + i_4 \geq 1, \quad i_2 + i_3 + i_4 < n$
- $i_1 + mi_2 + i_3 + mi_4 + i_5$ prim für alle $m \geq 0$.

Sei $m = i_1 + i_3 + i_5$. Dann kann $i_1 + mi_2 + i_3 + mi_4 + i_5 = (i_1 + i_3 + i_5)(1 + i_2 + i_4)$ nicht prim sein, da $i_1 + i_3 + i_5 = p - (i_2 + i_4) \geq p - n \geq 2$ und $1 + i_2 + i_4 \geq 2$.

Also $uv^m wx^m y \notin L_1$. Widerspruch.

Also kann L_1 nicht kontextfrei sein.