LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Formale Sprachen und Komplexität
Sommersemester 2019

Kontextfreie Sprachen: Teil I

Prof. Dr. David Sabel

LFE Theoretische Informatik

Letzte Änderung der Folien: 6. Juni 2019

Inhaltsübersicht

- Operationen auf Kontextfreien Grammatiken
- Normalformen für CFGs: Chomsky-Normalform und Greibach-Normalform
- Widerlegen der Kontextfreiheit: Pumping-Lemma

Lemma (Pumping-Lemma für CFLs)

Sei L eine kontextfreie Sprache. Dann gibt es eine Zahl $n \in \mathbb{N}_{>0}$, sodass jedes Wort $z \in L$, das Mindestlänge n hat (d. h. $|z| \geq n$), als $z = uvwxy$ geschrieben werden kann, so dass gilt:

- $|vx| \geq 1$
- $|vwx| \leq n$
- für alle $i \geq 0$: $uv^iwx^iy \in L$.

- Abschlusseigenschaften kontextfreier Sprachen

Kontextfreie Sprachen

Zur Erinnerung:

- Kontextfreie Sprachen (CFLs) werden von einer kontextfreien Grammatik (CFG) erzeugt
- Das sind die Typ 2-Grammatiken
- Bedingung: Alle linken Seiten der Produktionen bestehen aus genau einer Variablen.

Motivation

- Kontextfreie Sprachen sind insbesondere nützlich um Sprachen mit Klammerungen zu beschreiben
- Die Syntax von Programmiersprachen wird meist mit einer kontextfreien Grammatik angegeben.

Beispiele:

- $G = (\{E, M, Z\}, \{+, *, (,)\} \cup \{0, \ldots, 9\}, P, E)$ mit

$$\begin{aligned} P = \{ & E \to M \mid E + M, \\ & M \to Z \mid M * Z, \\ & Z \to N \mid (E), \\ & N \to 1D \mid \ldots \mid 9D, \\ & D \to 0D \mid \ldots \mid 9D \mid \varepsilon \} \end{aligned}$$

- $L = \{a^j b^j \mid j \in \mathbb{N}\}$ ist kontextfrei: Produktionen $\{S \to \varepsilon \mid T, T \to aTb \mid ab\}$ mit S als Startsymbol erzeugen L

Einfache Operationen auf CFGs

- Wir definieren Operationen auf CFGs, die die erzeugte Sprache unverändert lassen
- Die Operationen werden später (als Hilfsmittel) wiederverwendet.

Inlining von Produktionen

Lemma (Inlining von Produktionen)

Sei $G = (V, \Sigma, P, S)$ eine CFG und

- $A \to uBv \in P$,
- $B \to w_1 \mid \ldots \mid w_n$ alle Regeln mit B als linker Seite

und sei $G' = (V, \Sigma, P \setminus \{A \to uBv\} \cup \{A \to uw_1v \mid \ldots \mid uw_nv\}, S)$.

Dann erzeugen G und G' dieselbe Sprache, d. h. $L(G) = L(G')$.

Inlining von Produktionen

Lemma (Inlining von Produktionen)

Sei $G = (V, \Sigma, P, S)$ eine CFG und

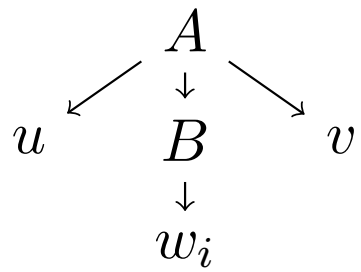
- $A \to uBv \in P$,
- $B \to w_1 \mid \ldots \mid w_n$ alle Regeln mit B als linker Seite

und sei $G' = (V, \Sigma, P \setminus \{A \to uBv\} \cup \{A \to uw_1v \mid \ldots \mid uw_nv\}, S)$.

Dann erzeugen G und G' dieselbe Sprache, d. h. $L(G) = L(G')$.

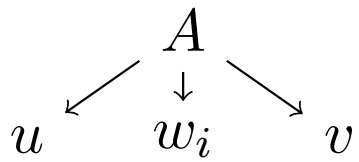
Beweis: Das folgt, indem die Syntaxbäume zur Ableitung mit G bzw. G' modfiziert werden:

tausche alle Baumabschnitte



(im Syntaxbaum mit G)

durch



(im Syntaxbaum mit G')

Sharing von Satzformen mit neuen Produktionen

Lemma (Sharing von Satzformen mit neuen Produktionen)

Sei G eine CFG mit $G = (V, \Sigma, P \cup \{A \to w_1 \cdots w_n\}, S)$.
Seien $B_1, \ldots, B_n$ neue Variablen (d. h. $V \cap \{B_1, \ldots, B_n\} = \emptyset$) und sei $G' = (V \cup \{B_1, \ldots, B_n\}, \Sigma, P \cup \{A \to B_1 \cdots B_n, B_1 \to w_1, \ldots, B_n \to w_n\}, S)$.
Dann gilt $L(G) = L(G')$.

Beweis:

"$\subseteq$": Konstruiere aus $S \Rightarrow_G^* w$ Ableitung $S \Rightarrow_{G'}^* w'$: Übersetze jeden Schritt $uAv \Rightarrow_G uw_1 \cdots w_n v$ in $uAv \Rightarrow_{G'} uB_1 \cdots B_n v \Rightarrow_{G'}^n uw_1 \cdots w_n v$.

"$\supseteq$": Betrachte den Syntaxbaum für $S \Rightarrow_{G'}^* w$.
Identifiziere die Anwendungen der Regeln $A \to B_1 \cdots B_n$, $B_i \to w_i$, und modifziere Syntaxbaum durch Anwendung der Regel $A \to w_1 \cdots w_n$.
Lese Ableitung $S \Rightarrow_G^* w$ ab.

Elimination der Links-Rekursion (1)

Definition (links- bzw. rechts-rekursive Produktion

Eine Produktion nennt man **links-rekursiv**, wenn sie von der Form

$$A \to Au$$

ist, und **rechts-rekursiv**, wenn sie von der Form

$$A \to uA$$

ist, wobei in beiden Fällen u eine Satzform ist.

Elimination der Links-Rekursion (2)

Lemma (Elimination von Links-Rekursion)

Sei $G = (V, \Sigma, P, S)$, $P = P' \cup \underbrace{\{A \to Au_1 \mid \cdots \mid Au_n \mid w_1 \mid \cdots \mid w_m\}}_{P''}$

eine CFG mit

- P'' sind alle Produktionen in P mit A als linker Seite und
- die Satzformen $w_1, \ldots, w_m$ beginnen alle nicht mit A.

Es gilt $L(G) = L(G')$ für $G' = (V \cup \{B\}, \Sigma, P' \cup P''', S)$ mit B neue Variable und $P''' = \{A \to w_1 B \mid \cdots \mid w_m B \mid w_1 \mid \cdots \mid w_m,$
$B \to u_1 \mid \ldots \mid u_n \mid u_1 B \mid \ldots \mid u_n B\}$

Beweis: Ausführlicher Beweis im Skript
Wesentliche Idee:
Ersetze Linksableitungsschritte mit G: $x_1 A x_2 \Rightarrow_G^* x_1 w_r u_{f(1)} \cdots u_{f(k)} x_2$
durch Rechtsableitungsschritte mit G': $x_1 A x_2 \Rightarrow_{G'}^* x_1 w_r u_{f(1)} \cdots u_{f(k)} x_2$

Normalformen

- Normalformen von Grammatiken fordern eine spezielle Form der Produktionen
- Nützlich, wenn man Grammatiken analysiert oder Algorithmen auf Grammatiken formuliert
- Man muss dann nur diese Form (statt aller erlaubten) von Produktionen betrachten
- Wir betrachten zwei Normalformen
  - Chomsky-Normalform
  - Greibach-Normalform

Die Chomsky-Normalform

Definition (Chomsky-Normalform)

Eine CFG $G = (V, \Sigma, P, S)$ mit $\varepsilon \notin L(G)$ ist in **Chomsky-Normalform**, wenn für $A \to w \in P$ gilt: $w = a \in \Sigma$ oder $w = BC$ mit $B, C \in V$.

Die Chomsky-Normalform

Definition (Chomsky-Normalform)

Eine CFG $G = (V, \Sigma, P, S)$ mit $\varepsilon \notin L(G)$ ist in **Chomsky-Normalform**, wenn für $A \to w \in P$ gilt: $w = a \in \Sigma$ oder $w = BC$ mit $B, C \in V$.

Beispiel:

- Die CFG $G = (\{A\}, \{(,), [,]\}, \{A \to (A) \mid () \mid [A] \mid [] \mid A\,A\}, A)$ ist **nicht** in Chomsky-Normalform
  (nur die Produktion $A \to A\,A$ passt zum vorgeschriebenen Format).
- Die CFG $G' = (\{A, B, C, D, E, F, G\}, \{(,), [,]\}, P, A)$ mit

$$P = \{A \to BF \mid BC \mid DG \mid DE \mid AA,\\ B \to (, C \to), D \to [, E \to], F \to AC, G \to AE\}$$

  ist in Chomsky-Normalform (und erzeugt die gleiche Sprache wie G).

Eigenschaften der Chomsky-Normalform

Sei G eine CFG in Chomsky-Normalform, dann gilt:

- Syntaxbäume zu Ableitungen mit G sind immer **Binärbäume**
- Ableitungen eines Worts $w \in L(G)$ bestehen immer genau aus $2 \cdot |w| - 1$ Ableitungsschritten.

Beispiel:
$G' = (\{A, B, C, D, E, F, G\}, \{(,), [,]\}, P, A)$ mit

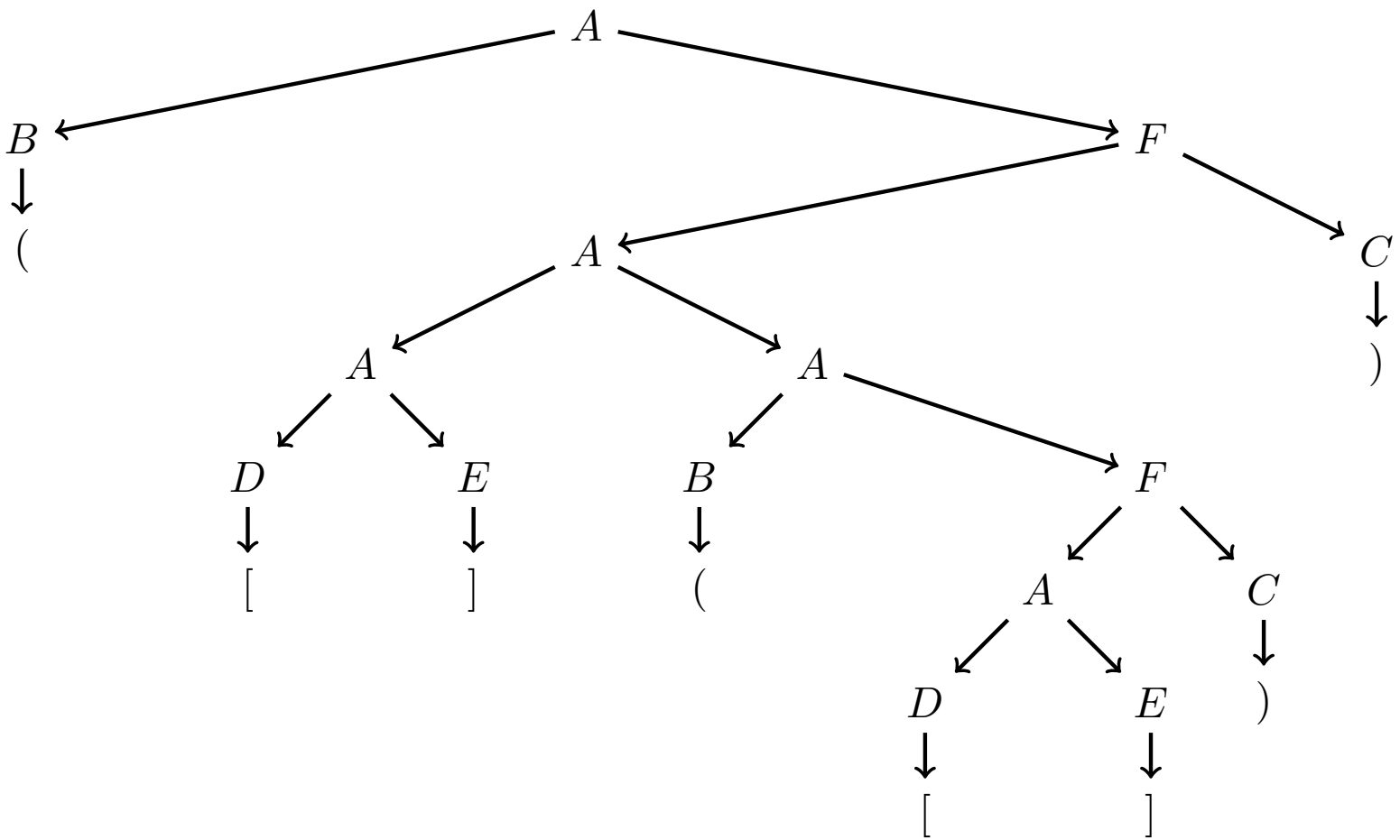
$$P = \{A \to BF \mid BC \mid DG \mid DE \mid AA,$$
$$B \to (, C \to), D \to [, E \to], F \to AC, G \to AE\}$$

Ableitung von $([]([]))$:

$$\begin{aligned} A &\Rightarrow BF \Rightarrow (F \Rightarrow (AC \Rightarrow (AAC \Rightarrow (DEAC \Rightarrow ([EAC \\ &\Rightarrow ([]AC \Rightarrow ([]BFC \Rightarrow ([](FC \Rightarrow ([](ACC \Rightarrow ([](DECC \\ &\Rightarrow ([]([ECC \Rightarrow ([]([]CC \Rightarrow ([]([])C \Rightarrow ([]([])) \end{aligned}$$

Eigenschaften der Chomsky-Normalform (2)

Der Syntaxbaum dazu:



Herstellen der Chomsky-Normalform

Jede CFG (mit $\varepsilon \notin L(G)$) kann in Chomsky-Normalform gebracht werden. Das Verfahren geht in mehreren Schritten vor:

1. Entfernen von ε-Produktionen (kennen wir bereits)
2. Entfernen von Einheitsproduktionen (Produktionen $A \to B$)
3. Sharen aller Terminale a in rechten Seiten, die nicht nur aus a bestehen durch neue Produktionen $A \to a$
4. Alle Produktionen $A \to B_1 \cdots B_m$ mit $m > 2$ in mehrere zerlegen: $A \to B_1 C_1, C_1 \to B_2 C_3, \ldots, C_{m-2} \to B_{m-1} B_m$

Entfernen von Einheitsproduktionen

- Intuitiv ist klar, dass $A \to B$ entfernt werden kann:
  Wenn erst $A \to B$, dann $B \to w$ angewendet wird, kann man auch gleich $A \to w$ anwenden.
- Algorithmisch zu beachten:
  - Eliminiere in der richtigen Reihenfolge:

    Wenn $A \to B$ und $B \to C$, dann ist Ersetzen von $A \to B$ durch $A \to C$ nicht zielführend.
  - Zyklen $A \to B$ und $B \to A$ müssen vorher entfernt werden!

Algorithmus 5: Entfernen von Einheitsproduktionen

Eingabe: Eine CFG $G = (V, \Sigma, P, S)$

Beginn

Erzeuge ger. Graph $D = (V, E)$, mit $(A, B) \in E$ für jede Einheitsp. $A \to B \in P$;

solange *es einen Zyklus* $(A_1, A_2), \ldots, (A_{n-1}, A_n), (A_n, A_1) \in E$ *gibt* **tue**

$P := P \setminus \{A_1 \to A_2, \ldots, A_{n-1} \to A_n, A_n \to A_n\}$; /* entferne zykl. Regeln */

$P := P[A_1/A_2, \ldots, A_1/A_n]$; /* ersetze alle Vorkommen von A_i durch A_1 für $i = 2, \ldots, n$ */

$V := V \setminus \{A_2, A_3, \ldots, A_n\}$; /* lösche $A_2, \ldots, A_n$ */

$S := S[A_1/A_2, \ldots, A_1/A_n]$; /* ersetze Startsymbol durch A_1, falls es A_i, $2 \leq i \leq n$ war */

$E := E \setminus \{(A_1, A_2), \ldots, (A_{n-1}, A_n), (A_n, A_1)\}$; /* entferne Zyklus aus Graph */

$E := E[A_1/A_2, \ldots, A_1/A_n]$; /* ersetze A_i durch A_1 für $i = 2, \ldots, n$ in anderen Kanten */

Sortiere D topologisch und nummeriere die Variablen in V durch (und benenne entsprechend in E, P, S um), so dass gilt: $A_i \to A_j$ impliziert $i < j$;

Sei $V = \{A_1, \ldots, A_k\}$;

für *i=k bis* 1 **tue**

wenn $A_i \to A_j \in P$ **dann**

seien $A_j \to w_1, \ldots, A_j \to w_m$ alle Produktionen mit A_j als linker Seite;

$P := P \cup \{A_i \to w_1, \ldots, A_i \to w_m\}$;

$P := P \setminus \{A_i \to A_j\}$;

Gib die so entstandene Grammatik als G' aus;

Korrektheit von Algorithmus 5 (1)

Satz

Algorithmus 5 berechnet bei Eingabe einer CFG G mit $\varepsilon \notin L(G)$ eine CFG G', die keine Einheitsproduktionen hat, sodass gilt $L(G) = L(G')$. Wenn G keine ε-Produktionen hat, dann hat auch G' keine ε-Produktionen.

Beweis: Wir zeigen:

1. Das Entfernen eines Zyklus verändert die erzeugte Sprache nicht
2. Das Entfernen einer Einheitsproduktion $A_i \to A_j$ in der rückwärts-laufenden für-Schleife ändert die erzeugte Sprache nicht.

   bereits gezeigt, da die Operation „Inlining von Produktionen“ korrekt.

3. Der Algorithmus terminiert und führt nie Einheits- oder ε-Produktionen ein.

Korrektheit von Alg. 5: Entfernen von Zyklen ist korrekt

Beweisskizze (ausführlicher Beweis im Skript):

- Habe $G = (V, \Sigma, P, S)$ Zyklus $A_1 \to A_2, \ldots, A_{n-1} \to A_n, A_n \to A_1$
- Sei $G' = (V', \Sigma, P', S')$ die Grammatik nach Entfernen des Zyklus.
- Sei σ die Substitution $\{A_i \mapsto A_1 \mid i \in \{2, \ldots, n\}\}$.
- „$L(G) \subseteq L(G')$“:
  - Zeige mit Induktion über n, dass für alle $w_1, w_2 \in (\Sigma \cup V)^*$ gilt: Wenn $w_1 \Rightarrow_G^n w_2$, dann $\sigma(w_1) \Rightarrow_{G'}^* \sigma(w_2)$.
- „$L(G') \subseteq L(G)$“:
  - Zeige mit Induktion über n, dass für $w \in (\Sigma \cup V')^*$ und $w_0 \in \Sigma^*$ gilt: Wenn $w \Rightarrow_{G'}^n w_0 \in \Sigma^*$. dann $w \Rightarrow_G^* w_0$.

Korrektheit von Alg. 5: Terminierung

Algorithmus 5 terminiert, denn

- Die Solange-Schleife terminiert, da jede Iteration die Anzahl der Zyklen strikt verkleinert.
- Die für-Schleife terminiert offensichtlich.

Es werden keine Einheitsproduktionen eingeführt:

- Da die Produktionen topologisch sortiert behandelt werden: Wenn $A_i \to A_j$ entfernt wird, wurden **vorher** alle Einheitsproduktionen $A_j \to A_k$ entfernt. D.h. zu diesem Zeitpunkt gilt: für alle Produktionen $A_j \to w$ besteht w nicht nur aus einer Variablen.

Es werden keine ε-Produktionen eingeführt:

- offensichtlich.

Algorithmus 6: Herstellen der Chomsky-NF

Eingabe: CFG G mit $\varepsilon \notin L(G)$
Ausgabe: CFG G' in Chomsky-Normalform mit $L(G) = L(G')$
Beginn

Entferne die ε-Produktionen in G mit Algorithmus 1 und entferne anschließend die Einheitsproduktionen mit Algorithmus 5;

Sei $G' = (V', \Sigma, P', S')$ die entstandene Grammatik;

für alle $a \in \Sigma$ **tue**

/* Führe neue Variable A_a für a ein, und ersetze Vorkommen von a durch das Nichtterminal */

$$G' := (V' \cup \{A_a\}, \Sigma, \{A \to w[A_a/a] \mid A \to w \in P' \text{ und } |w| > 1\} \cup \{A \to w \mid A \in w \in P' \text{ und } |w| = 1\} \cup \{A_a \to a\}, S)$$

/* Nun sind alle Regeln von der Form $A \to a$ oder $A \to B_1 \cdots B_m$ mit $m \geq 2$ */

für alle $A \to B_1 \cdots B_m \in P'$ *mit* $m > 2$ **tue**

Seien $C_1, \ldots, C_{m-2}$ neue Variablen;

$V' := V' \cup \{C_1, \ldots, C_{m-2}\}$;

/* Ersetze in P' die Produktion $A \to B_1 \cdots B_m$ durch neue Regeln */

$$P' := (P' \setminus \{A \to B_1 \cdots B_m\}) \cup \{A \to B_1 C_1\} \cup \{C_i \to B_{i+1} C_{i+1} \mid \text{für } i = 1, \ldots, m-3\} \cup \{C_{m-2} \to B_{m-1} B_m\};$$

Theorem

Für CFGs G mit $\varepsilon \notin L(G)$ berechnet Algorithmus 6 eine äquivalente CFG in Chomsky-Normalform.

Beweis:

- Die Schritte „Entferne der ε-Produktionen und Entfernen von Einheitsproduktionen haben wir als korrekt gezeigt.
- Die Schritte „Einführen von Produktionen $A \to a$“ und Behandlung von $A \to B_1 \cdots B_m$ sind Instanzen der korrekten Operation „Sharing von Satzformen mit neuen Variablen“
- Verifiziere, dass danach alle Produktionen die gewünschte Form haben.

Beispiel: Chomsky-Normalform berechnen

$G_0 = (\{A, B, C, D, S\}, \{0, 1\}, P_0, S)$ mit

$$P_0 = \{S \to 1A,\ \ A \to AB,\ \ A \to DA,\ \ A \to \varepsilon,\\ B \to 0,\ \ B \to 1,\ \ C \to AAA,\ \ D \to 1AC\}.$$

Schritt 1: Entfernen der ε-Produktionen

$G_0 = (\{A, B, C, D, S\}, \{0, 1\}, P_0, S)$ mit

$$P_0 = \{S \to 1A,\ \ A \to AB,\ \ A \to DA,\ \ A \to \varepsilon,\ \ B \to 0,\\ B \to 1,\ \ C \to AAA,\ \ D \to 1AC\}.$$

Schritt 1: Entfernen der ε-Produktionen

$G_0 = (\{A, B, C, D, S\}, \{0, 1\}, P_0, S)$ mit

$$P_0 = \{S \to 1A,\ \ A \to AB,\ \ A \to DA,\ \ A \to \varepsilon,\ \ B \to 0,\\ B \to 1,\ \ C \to AAA,\ \ D \to 1AC\}.$$

- Menge W der Variablen, die ε herleiten:

  $W = \{A, C\}$ da $A \to \varepsilon$ und $C \to AAA$

Schritt 1: Entfernen der ε-Produktionen

$G_0 = (\{A, B, C, D, S\}, \{0, 1\}, P_0, S)$ mit

$$P_0 = \{S \to 1A,\ \ A \to AB,\ \ A \to DA,\ \ A \to \varepsilon,\ \ B \to 0,\\ B \to 1,\ \ C \to AAA,\ \ D \to 1AC\}.$$

- Menge W der Variablen, die ε herleiten:

  $W = \{A, C\}$ da $A \to \varepsilon$ und $C \to AAA$

- Starte mit

  $G_1 = (\{A, B, C, D, S\}, \{0, 1\}, P_1, S)$
  $P_1 = \{S \to 1A,\ \ A \to AB,\ \ A \to DA,\ \ B \to 0,\\ B \to 1,\ \ C \to AAA,\ \ D \to 1AC\}.$

Schritt 1: Entfernen der ε-Produktionen

$G_0 = (\{A, B, C, D, S\}, \{0, 1\}, P_0, S)$ mit

$$P_0 = \{S \to 1A,\ \ A \to AB,\ \ A \to DA,\ \ A \to \varepsilon,\ \ B \to 0,\\ B \to 1,\ \ C \to AAA,\ \ D \to 1AC\}.$$

- Menge W der Variablen, die ε herleiten:

  $W = \{A, C\}$ da $A \to \varepsilon$ und $C \to AAA$
- Starte mit

  $G_1 = (\{A, B, C, D, S\}, \{0, 1\}, P_1, S)$
  $P_1 = \{S \to 1A,\ \ A \to AB,\ \ A \to DA,\ \ B \to 0,\\ B \to 1,\ \ C \to AAA,\ \ D \to 1AC\}.$
- Hinzufügen von Produktionen für Vorkommen von A und C

  $P_1 = \{S \to 1A,\ \ S \to 1,\ \ A \to AB,\ \ A \to B,\ \ A \to DA,\ \ A \to D,\\ B \to 0,\ \ B \to 1,\ \ C \to AAA,\ \ C \to AA,\ \ C \to A,\\ D \to 1AC,\ \ D \to 1A,\ \ D \to 1C,\ \ D \to 1\}.$

Schritt 2: Entfernen der Einheitsproduktionen (1)

$$P_1 = \{S \to 1A,\ \ S \to 1,\ \ A \to AB,\ \ A \to B,\ \ A \to DA,\ \ A \to D,$$
$$B \to 0,\ \ B \to 1\ \ C \to AAA,\ \ C \to AA,\ \ C \to A,$$
$$D \to 1AC,\ \ D \to 1A,\ \ D \to 1C,\ \ D \to 1\}.$$

- Der gerichtete Graph ist
  $\mathsf{D} = (\{S, A, B, C, D\}, \{(A, B), (A, D), (C, A)\})$.
- Es gibt keine Zyklen, wir erhalten $P_2 = P_1$
- Topologisches Sortieren und Umbenennen der Variablen, sodass „$A_i \to A_j$ impliziert $i < j$“ gilt, erfordert eine Umbenennung, welche die Beziehungen „$A < B$“, „$A < D$“, „$C < A$“ erzeugt.
- Wir wählen Umbenennung ρ mit $\rho(C) = A_1$, $\rho(A) = A_2$, $\rho(B) = A_3$, $\rho(D) = A_4$, $\rho(S) = A_5$.
- Das liefert uns $G_3 = (\{A_1, A_2, A_3, A_4, A_5\}, \Sigma, P_3, A_5)$ mit

$$P_3 = \{A_5 \to 1A_2,\ \ A_5 \to 1,\ \ A_2 \to A_2A_3,\ \ A_2 \to A_3,\ \ A_2 \to A_4A_2,$$
$$A_2 \to A_4,\ \ A_3 \to 0,\ \ A_3 \to 1,\ \ A_1 \to A_2A_2A_2,\ \ A_1 \to A_2A_2,$$
$$A_1 \to A_2,\ \ A_4 \to 1A_2A_1,\ \ A_4 \to 1A_2,\ \ A_4 \to 1A_1,\ \ A_4 \to 1\}.$$

Schritt 2: Entfernen der Einheitsproduktionen (1)

$$P_1 = \{S \to 1A,\ \ S \to 1,\ \ A \to AB,\ \ A \to B,\ \ A \to DA,\ \ A \to D,$$
$$B \to 0,\ \ B \to 1\ \ C \to AAA,\ \ C \to AA,\ \ C \to A,$$
$$D \to 1AC,\ \ D \to 1A,\ \ D \to 1C,\ \ D \to 1\}.$$

- Der gerichtete Graph ist
  $\mathsf{D} = (\{S, A, B, C, D\}, \{(A, B), (A, D), (C, A)\})$.
- Es gibt keine Zyklen, wir erhalten $P_2 = P_1$
- Topologisches Sortieren und Umbenennen der Variablen, sodass „$A_i \to A_j$ impliziert $i < j$“ gilt, erfordert eine Umbenennung, welche die Beziehungen „$A < B$“, „$A < D$“, „$C < A$“ erzeugt.
- Wir wählen Umbenennung ρ mit $\rho(C) = A_1$, $\rho(A) = A_2$, $\rho(B) = A_3$, $\rho(D) = A_4$, $\rho(S) = A_5$.
- Das liefert uns $G_3 = (\{A_1, A_2, A_3, A_4, A_5\}, \Sigma, P_3, A_5)$ mit

$$P_3 = \{A_5 \to 1A_2,\ \ A_5 \to 1,\ \ A_2 \to A_2A_3,\ \ A_2 \to A_3,\ \ A_2 \to A_4A_2,$$
$$A_2 \to A_4,\ \ A_3 \to 0,\ \ A_3 \to 1,\ \ A_1 \to A_2A_2A_2,\ \ A_1 \to A_2A_2,$$
$$A_1 \to A_2,\ \ A_4 \to 1A_2A_1,\ \ A_4 \to 1A_2,\ \ A_4 \to 1A_1,\ \ A_4 \to 1\}.$$

Schritt 2: Entfernen der Einheitsproduktionen (1)

$$P_1 = \{S \to 1A,\ \ S \to 1,\ \ A \to AB,\ \ A \to B,\ \ A \to DA,\ \ A \to D,$$
$$B \to 0,\ \ B \to 1\ \ C \to AAA,\ \ C \to AA,\ \ C \to A,$$
$$D \to 1AC,\ \ D \to 1A,\ \ D \to 1C,\ \ D \to 1\}.$$

- Der gerichtete Graph ist
  $\mathsf{D} = (\{S, A, B, C, D\}, \{(A, B), (A, D), (C, A)\})$.
- Es gibt keine Zyklen, wir erhalten $P_2 = P_1$
- Topologisches Sortieren und Umbenennen der Variablen, sodass „$A_i \to A_j$ impliziert $i < j$“ gilt, erfordert eine Umbenennung, welche die Beziehungen „$A < B$“, „$A < D$“, „$C < A$“ erzeugt.
- Wir wählen Umbenennung ρ mit $\rho(C) = A_1$, $\rho(A) = A_2$, $\rho(B) = A_3$, $\rho(D) = A_4$, $\rho(S) = A_5$.
- Das liefert uns $G_3 = (\{A_1, A_2, A_3, A_4, A_5\}, \Sigma, P_3, A_5)$ mit

$$P_3 = \{A_5 \to 1A_2,\ \ A_5 \to 1,\ \ A_2 \to A_2A_3,\ \ A_2 \to A_3,\ \ A_2 \to A_4A_2,$$
$$A_2 \to A_4,\ \ A_3 \to 0,\ \ A_3 \to 1,\ \ A_1 \to A_2A_2A_2,\ \ A_1 \to A_2A_2,$$
$$A_1 \to A_2,\ \ A_4 \to 1A_2A_1,\ \ A_4 \to 1A_2,\ \ A_4 \to 1A_1,\ \ A_4 \to 1\}.$$

Schritt 2: Entfernen der Einheitsproduktionen (1)

$$P_1 = \{S \to 1A,\ \ S \to 1,\ \ A \to AB,\ \ A \to B,\ \ A \to DA,\ \ A \to D,$$
$$B \to 0,\ \ B \to 1\ \ C \to AAA,\ \ C \to AA,\ \ C \to A,$$
$$D \to 1AC,\ \ D \to 1A,\ \ D \to 1C,\ \ D \to 1\}.$$

- Der gerichtete Graph ist
  $\mathsf{D} = (\{S, A, B, C, D\}, \{(A, B), (A, D), (C, A)\})$.
- Es gibt keine Zyklen, wir erhalten $P_2 = P_1$
- Topologisches Sortieren und Umbenennen der Variablen, sodass „$A_i \to A_j$ impliziert $i < j$“ gilt, erfordert eine Umbenennung, welche die Beziehungen „$A < B$“, „$A < D$“, „$C < A$“ erzeugt.
- Wir wählen Umbenennung ρ mit $\rho(C) = A_1$, $\rho(A) = A_2$, $\rho(B) = A_3$, $\rho(D) = A_4$, $\rho(S) = A_5$.
- Das liefert uns $G_3 = (\{A_1, A_2, A_3, A_4, A_5\}, \Sigma, P_3, A_5)$ mit

$$P_3 = \{A_5 \to 1A_2,\ \ A_5 \to 1,\ \ A_2 \to A_2A_3,\ \ A_2 \to A_3,\ \ A_2 \to A_4A_2,$$
$$A_2 \to A_4,\ \ A_3 \to 0,\ \ A_3 \to 1,\ \ A_1 \to A_2A_2A_2,\ \ A_1 \to A_2A_2,$$
$$A_1 \to A_2,\ \ A_4 \to 1A_2A_1,\ \ A_4 \to 1A_2,\ \ A_4 \to 1A_1,\ \ A_4 \to 1\}.$$

Schritt 2: Entfernen der Einheitsproduktionen (1)

$$P_1 = \{S \to 1A,\ \ S \to 1,\ \ A \to AB,\ \ A \to B,\ \ A \to DA,\ \ A \to D,$$
$$B \to 0,\ \ B \to 1\ \ C \to AAA,\ \ C \to AA,\ \ C \to A,$$
$$D \to 1AC,\ \ D \to 1A,\ \ D \to 1C,\ \ D \to 1\}.$$

- Der gerichtete Graph ist
  $\mathsf{D} = (\{S, A, B, C, D\}, \{(A, B), (A, D), (C, A)\})$.
- Es gibt keine Zyklen, wir erhalten $P_2 = P_1$
- Topologisches Sortieren und Umbenennen der Variablen, sodass „$A_i \to A_j$ impliziert $i < j$“ gilt, erfordert eine Umbenennung, welche die Beziehungen „$A < B$“, „$A < D$“, „$C < A$“ erzeugt.
- Wir wählen Umbenennung ρ mit $\rho(C) = A_1$, $\rho(A) = A_2$, $\rho(B) = A_3$, $\rho(D) = A_4$, $\rho(S) = A_5$.
- Das liefert uns $G_3 = (\{A_1, A_2, A_3, A_4, A_5\}, \Sigma, P_3, A_5)$ mit

$$P_3 = \{A_5 \to 1A_2,\ \ A_5 \to 1,\ \ A_2 \to A_2A_3,\ \ A_2 \to A_3,\ \ A_2 \to A_4A_2,$$
$$A_2 \to A_4,\ \ A_3 \to 0,\ \ A_3 \to 1,\ \ A_1 \to A_2A_2A_2,\ \ A_1 \to A_2A_2,$$
$$A_1 \to A_2,\ \ A_4 \to 1A_2A_1,\ \ A_4 \to 1A_2,\ \ A_4 \to 1A_1,\ \ A_4 \to 1\}.$$

Schritt 2: Entfernen der Einheitsproduktionen (1)

$$P_1 = \{S \to 1A,\ \ S \to 1,\ \ A \to AB,\ \ A \to B,\ \ A \to DA,\ \ A \to D,$$
$$B \to 0,\ \ B \to 1\ \ C \to AAA,\ \ C \to AA,\ \ C \to A,$$
$$D \to 1AC,\ \ D \to 1A,\ \ D \to 1C,\ \ D \to 1\}.$$

- Der gerichtete Graph ist
  $\mathsf{D} = (\{S, A, B, C, D\}, \{(A, B), (A, D), (C, A)\})$.
- Es gibt keine Zyklen, wir erhalten $P_2 = P_1$
- Topologisches Sortieren und Umbenennen der Variablen, sodass „$A_i \to A_j$ impliziert $i < j$“ gilt, erfordert eine Umbenennung, welche die Beziehungen „$A < B$“, „$A < D$“, „$C < A$“ erzeugt.
- Wir wählen Umbenennung ρ mit $\rho(C) = A_1$, $\rho(A) = A_2$, $\rho(B) = A_3$, $\rho(D) = A_4$, $\rho(S) = A_5$.
- Das liefert uns $G_3 = (\{A_1, A_2, A_3, A_4, A_5\}, \Sigma, P_3, A_5)$ mit

$$P_3 = \{A_5 \to 1A_2,\ \ A_5 \to 1,\ \ A_2 \to A_2A_3,\ \ A_2 \to A_3,\ \ A_2 \to A_4A_2,$$
$$A_2 \to A_4,\ \ A_3 \to 0,\ \ A_3 \to 1,\ \ A_1 \to A_2A_2A_2,\ \ A_1 \to A_2A_2,$$
$$A_1 \to A_2,\ \ A_4 \to 1A_2A_1,\ \ A_4 \to 1A_2,\ \ A_4 \to 1A_1,\ \ A_4 \to 1\}.$$

Schritt 2: Entfernen der Einheitsproduktionen (2)

$$P_3 = \{A_5 \to 1A_2,\ \ A_5 \to 1,\ \ A_2 \to A_2A_3,\ \ A_2 \to A_3,\ \ A_2 \to A_4A_2,\ \ A_2 \to A_4,\ \ A_3 \to 0,\ \ A_3 \to 1,\ \ A_1 \to A_2A_2A_2,\ \ A_1 \to A_2A_2,\ \ A_1 \to A_2,\ \ A_4 \to 1A_2A_1,\ \ A_4 \to 1A_2,\ \ A_4 \to 1A_1,\ \ A_4 \to 1\}.$$

- Nun läuft die Für-Schleife für i von 5 bis 1:
  - Für $i = 5, i = 4, i = 3$ gibt es jeweils keine Produktion der Form $A_i \to A_j$.
  - Für $i = 2$ wird $A_2 \to A_3$ ersetzt durch $A_2 \to 0,\ \ A_2 \to 1$, und es wird $A_2 \to A_4$ ersetzt durch $A_2 \to 1A_2A_1,\ \ A_2 \to 1A_2,\ \ A_2 \to 1A_1$ und $A_2 \to 1$. Danach ist

    $$P_4 = \{A_5 \to 1A_2,\ \ A_5 \to 1,\ \ A_2 \to A_2A_3,\ \ A_2 \to 0,\ \ A_2 \to 1,\ \ A_2 \to A_4A_2,\ \ A_2 \to 1A_2A_1,\ \ A_2 \to 1A_2,\ \ A_2 \to 1A_1,\ \ A_3 \to 0,\ \ A_3 \to 1,\ \ A_1 \to A_2A_2A_2,\ \ A_1 \to A_2A_2,\ \ A_1 \to A_2,\ \ A_4 \to 1A_2A_1,\ \ A_4 \to 1A_2,\ \ A_4 \to 1A_1,\ \ A_4 \to 1\}.$$

  - Für $i = 1$ wird $A_1 \to A_2$ ersetzt durch $A_1 \to A_2A_3$, $A_1 \to 0$, $A_1 \to 1$, $A_1 \to A_4A_2$, $A_1 \to 1A_2A_1$, $A_1 \to 1A_2$ und $A_1 \to 1A_1$.

Schritt 2: Entfernen der Einheitsproduktionen (2)

$$P_3 = \{A_5 \to 1A_2,\ \ A_5 \to 1,\ \ A_2 \to A_2A_3,\ \ A_2 \to A_3,\ \ A_2 \to A_4A_2,\ \ A_2 \to A_4,\ \ A_3 \to 0,\ \ A_3 \to 1,\ \ A_1 \to A_2A_2A_2,\ \ A_1 \to A_2A_2,\ \ A_1 \to A_2,\ \ A_4 \to 1A_2A_1,\ \ A_4 \to 1A_2,\ \ A_4 \to 1A_1,\ \ A_4 \to 1\}.$$

- Nun läuft die Für-Schleife für i von 5 bis 1:
  - Für $i = 5, i = 4, i = 3$ gibt es jeweils keine Produktion der Form $A_i \to A_j$.
  - Für $i = 2$ wird $A_2 \to A_3$ ersetzt durch $A_2 \to 0,\ \ A_2 \to 1$, und es wird $A_2 \to A_4$ ersetzt durch $A_2 \to 1A_2A_1,\ \ A_2 \to 1A_2,\ \ A_2 \to 1A_1$ und $A_2 \to 1$. Danach ist

    $$P_4 = \{A_5 \to 1A_2,\ \ A_5 \to 1,\ \ A_2 \to A_2A_3,\ \ A_2 \to 0,\ \ A_2 \to 1,\ \ A_2 \to A_4A_2,\ \ A_2 \to 1A_2A_1,\ \ A_2 \to 1A_2,\ \ A_2 \to 1A_1,\ \ A_3 \to 0,\ \ A_3 \to 1,\ \ A_1 \to A_2A_2A_2,\ \ A_1 \to A_2A_2,\ \ A_1 \to A_2,\ \ A_4 \to 1A_2A_1,\ \ A_4 \to 1A_2,\ \ A_4 \to 1A_1,\ \ A_4 \to 1\}.$$

  - Für $i = 1$ wird $A_1 \to A_2$ ersetzt durch $A_1 \to A_2A_3$, $A_1 \to 0$, $A_1 \to 1$, $A_1 \to A_4A_2$, $A_1 \to 1A_2A_1$, $A_1 \to 1A_2$ und $A_1 \to 1A_1$.

Schritt 2: Entfernen der Einheitsproduktionen (2)

$$P_3 = \{A_5 \to 1A_2,\ \ A_5 \to 1,\ \ A_2 \to A_2A_3,\ \ A_2 \to A_3,\ \ A_2 \to A_4A_2,\ \ A_2 \to A_4,\ \ A_3 \to 0,\ \ A_3 \to 1,\ \ A_1 \to A_2A_2A_2,\ \ A_1 \to A_2A_2,\ \ A_1 \to A_2,\ \ A_4 \to 1A_2A_1,\ \ A_4 \to 1A_2,\ \ A_4 \to 1A_1,\ \ A_4 \to 1\}.$$

- Nun läuft die Für-Schleife für i von 5 bis 1:
  - Für $i = 5, i = 4, i = 3$ gibt es jeweils keine Produktion der Form $A_i \to A_j$.
  - Für $i = 2$ wird $A_2 \to A_3$ ersetzt durch $A_2 \to 0,\ \ A_2 \to 1$, und es wird $A_2 \to A_4$ ersetzt durch $A_2 \to 1A_2A_1,\ \ A_2 \to 1A_2,\ \ A_2 \to 1A_1$ und $A_2 \to 1$. Danach ist

    $$P_4 = \{A_5 \to 1A_2,\ \ A_5 \to 1,\ \ A_2 \to A_2A_3,\ \ A_2 \to 0,\ \ A_2 \to 1,\ \ A_2 \to A_4A_2,\ \ A_2 \to 1A_2A_1,\ \ A_2 \to 1A_2,\ \ A_2 \to 1A_1,\ \ A_3 \to 0,\ \ A_3 \to 1,\ \ A_1 \to A_2A_2A_2,\ \ A_1 \to A_2A_2,\ \ A_1 \to A_2,\ \ A_4 \to 1A_2A_1,\ \ A_4 \to 1A_2,\ \ A_4 \to 1A_1,\ \ A_4 \to 1\}.$$

  - Für $i = 1$ wird $A_1 \to A_2$ ersetzt durch $A_1 \to A_2A_3$, $A_1 \to 0$, $A_1 \to 1$, $A_1 \to A_4A_2$, $A_1 \to 1A_2A_1$, $A_1 \to 1A_2$ und $A_1 \to 1A_1$.

Schritt 2: Entfernen der Einheitsproduktionen (2)

$$P_3 = \{A_5 \to 1A_2,\ \ A_5 \to 1,\ \ A_2 \to A_2A_3,\ \ A_2 \to A_3,\ \ A_2 \to A_4A_2,\ \ A_2 \to A_4,\ \ A_3 \to 0,\ \ A_3 \to 1,\ \ A_1 \to A_2A_2A_2,\ \ A_1 \to A_2A_2,\ \ A_1 \to A_2,\ \ A_4 \to 1A_2A_1,\ \ A_4 \to 1A_2,\ \ A_4 \to 1A_1,\ \ A_4 \to 1\}.$$

- Nun läuft die Für-Schleife für i von 5 bis 1:
  - Für $i = 5, i = 4, i = 3$ gibt es jeweils keine Produktion der Form $A_i \to A_j$.
  - Für $i = 2$ wird $A_2 \to A_3$ ersetzt durch $A_2 \to 0,\ \ A_2 \to 1$, und es wird $A_2 \to A_4$ ersetzt durch $A_2 \to 1A_2A_1,\ \ A_2 \to 1A_2,\ \ A_2 \to 1A_1$ und $A_2 \to 1$. Danach ist

    $$P_4 = \{A_5 \to 1A_2,\ \ A_5 \to 1,\ \ A_2 \to A_2A_3,\ \ A_2 \to 0,\ \ A_2 \to 1,\ \ A_2 \to A_4A_2,\ \ A_2 \to 1A_2A_1,\ \ A_2 \to 1A_2,\ \ A_2 \to 1A_1,\ \ A_3 \to 0,\ \ A_3 \to 1,\ \ A_1 \to A_2A_2A_2,\ \ A_1 \to A_2A_2,\ \ A_1 \to A_2,\ \ A_4 \to 1A_2A_1,\ \ A_4 \to 1A_2,\ \ A_4 \to 1A_1,\ \ A_4 \to 1\}.$$

  - Für $i = 1$ wird $A_1 \to A_2$ ersetzt durch $A_1 \to A_2A_3$, $A_1 \to 0$, $A_1 \to 1$, $A_1 \to A_4A_2$, $A_1 \to 1A_2A_1$, $A_1 \to 1A_2$ und $A_1 \to 1A_1$.

Schritt 2: Entfernen der Einheitsproduktionen (3)

- Daher ist die Grammatik nach Entfernen der Einheitsproduktionen: $G_5 = (V_5, \Sigma, P_5, A_5)$ mit $V_5 = \{A_1, A_2, A_3, A_4, A_5\}$ und

$$
\begin{aligned}
P_5 = \{ & A_5 \to 1A_2,\ \ A_5 \to 1,\ \ A_2 \to A_2A_3,\ \ A_2 \to 0,\ \ A_2 \to 1,\\
& A_2 \to A_4A_2, A_2 \to 1A_2A_1,\ \ A_2 \to 1A_2,\ \ A_2 \to 1A_1,\ \ A_3 \to 0,\\
& A_3 \to 1,\ \ A_1 \to A_2A_2A_2,\ \ A_1 \to A_2A_2,\ \ A_1 \to A_2A_3,\ \ A_1 \to 0,\\
& A_1 \to 1,\ \ A_1 \to A_4A_2,\ \ A_1 \to 1A_2A_1,\ \ A_1 \to 1A_2,\ \ A_1 \to 1A_1,\\
& A_4 \to 1A_2A_1,\ \ A_4 \to 1A_2,\ \ A_4 \to 1A_1,\ \ A_4 \to 1\}.
\end{aligned}
$$

Schritt 3: Terminalsymbole durch neue Produktionen darstellen

Füge $B_0 \to 0$ und $B_1 \to 1$ hinzu und ersetze in rechten Seiten mit Wortlänge > 1:

$$
\begin{aligned}
P_6 = \{&B_0 \to 0,\ \ B_1 \to 1,\ \ A_5 \to B_1A_2,\ \ A_5 \to 1,\ \ A_2 \to A_2A_3,\ \ A_2 \to 0,\\
&A_2 \to 1,\ \ A_2 \to A_4A_2,\ \ A_2 \to B_1A_2A_1,\ \ A_2 \to B_1A_2,\\
&A_2 \to B_1A_1,\ \ A_3 \to 0,\ \ A_3 \to 1,\ \ A_1 \to A_2A_2A_2,\ \ A_1 \to A_2A_2,\\
&A_1 \to A_2A_3,\ \ A_1 \to 0,\ \ A_1 \to 1,\ \ A_1 \to A_4A_2,\ \ A_1 \to B_1A_2A_1,\\
&A_1 \to B_1A_2,\ \ A_1 \to B_1A_1,\ \ A_4 \to B_1A_2A_1,\\
&A_4 \to B_1A_2,\ \ A_4 \to B_1A_1,\ \ A_4 \to 1\}.
\end{aligned}
$$

Schritt 4: Rechte Seiten zerlegen

Zerlege rechte Seiten mit Wortlänge > 2:

Ergibt $G_7 = (V_7, \Sigma, P_7, A_5)$, wobei

$$V_7 = \{A_1, A_2, A_3, A_4, A_5, B_0, B_1, C_1, C_2, C_3, C_4\}$$

$$\begin{aligned} P_7 = \{&B_0 \to 0,\ \ B_1 \to 1,\ \ A_5 \to B_1A_2,\ \ A_5 \to 1,\ \ A_2 \to A_2A_3,\ \ A_2 \to 0,\\ &A_2 \to 1,\ \ A_2 \to A_4A_2,\ \ A_2 \to B_1C_1\ \ C_1 \to A_2A_1,\\ &A_2 \to B_1A_2,\ \ A_2 \to B_1A_1,\ \ A_3 \to 0,\ \ A_3 \to 1,\ \ A_1 \to A_2C_2,\\ &C_2 \to A_2A_2,\ \ A_1 \to A_2A_2,\ \ A_1 \to A_2A_3,\ \ A_1 \to 0,\ \ A_1 \to 1,\\ &A_1 \to A_4A_2,\ \ A_1 \to B_1C_3,\ \ C_4 \to A_2A_1,\ \ A_1 \to B_1A_2,\\ &A_1 \to B_1A_1,\ \ A_4 \to B_1C_4,\ \ C_4 \to A_2A_1,\ \ A_4 \to B_1A_2,\\ &A_4 \to B_1A_1,\ \ A_4 \to 1\}. \end{aligned}$$

Alle Schritte beendet, G_7 ist in Chomsky-Normalform

Greibach-Normalform

Definition (Greibach-Normalform)

Ein CFG $G = (V, \Sigma, P, S)$ mit $\varepsilon \notin L(G)$ ist in *Greibach-Normalform*, falls alle Produktionen in P von der Form $A \to aB_1B_2 \dots B_j$ mit $j \geq 0$, $A, B_1, \dots, B_j \in V$ und $a \in \Sigma$ sind.

Bemerkungen:

- benannt nach Sheila A. Greibach
- Reguläre Grammatiken sind Spezialfall der Greibach-NF: Dort ist nur $j = 0$ oder $j = 1$ erlaubt.

Algorithmus 7: Herstellen der Greibach-Normalform

Eingabe: CFG $G = (\{A_1, \ldots, A_n\}, \Sigma, P, A_i)$ in Chomsky-NF mit $\varepsilon \notin L(G)$
Ausgabe: CFG G' in Greibach-Normalform mit $L(G) = L(G')$
Beginn

- **für** $i = 1$ *bis* n **tue**
  - **für** $j = 1$ *bis* $i - 1$ **tue**
    - **für alle** $A_i \to A_j u \in P$ **tue**
      - Seien $A_j \to w_1 \mid \cdots \mid w_m$ alle Regeln in P mit A_j als linker Seite;
      - Ersetze $A_i \to A_j u$ durch $A_i \to w_1 u \mid \ldots \mid w_m u$ in P;
  - **wenn** $A_i \to A_i u \in P$ **dann**
    - Eliminiere die Regel mit der Operation „Elimination der Links-Rekursion“;
    - Sei B_i die dabei neu erzeugte Variable;
- **für** $i = n - 1$ *bis* 1 **tue**
  - **für alle** $A_i \to A_j u \in P$, $j > i$ **tue**
    - Seien $A_j \to w_1 \mid \cdots \mid w_m$ alle Regeln mit A_j als linker Seite;
    - Ersetze $A_i \to A_j u$ durch $A_i \to w_1 u \mid \cdots \mid w_m u$ in P;
- **für** $i = 1$ *bis* n **tue**
  - **für alle** $B_i \to A_j u \in P$ **tue**
    - Seien $A_j \to w_1 \mid \cdots \mid w_m$ alle Regeln mit A_j als linker Seite;
    - Ersetze $B_i \to A_j u$ durch $B_i \to w_1 u \mid \cdots \mid w_m u$ in P;

Algorithmus 7: Herstellen der Greibach-Normalform

Eingabe: CFG $G = (\{A_1, \ldots, A_n\}, \Sigma, P, A_i)$ in Chomsky-NF mit $\varepsilon \notin L(G)$
Ausgabe: CFG G' in Greibach-Normalform mit $L(G) = L(G')$
Beginn

Ziel der geschachtelten Für-Schleife: Es gibt $A_i \to A_j u$ nur für $j > i$

- **für** $i = 1$ *bis* n **tue**
  - **für** $j = 1$ *bis* $i - 1$ **tue**
    - **für alle** $A_i \to A_j u \in P$ **tue**
      - Seien $A_j \to w_1 \mid \cdots \mid w_m$ alle Regeln in P mit A_j als linker Seite;
      - Ersetze $A_i \to A_j u$ durch $A_i \to w_1 u \mid \ldots \mid w_m u$ in P;
  - **wenn** $A_i \to A_i u \in P$ **dann**
    - Eliminiere die Regel mit der Operation „Elimination der Links-Rekursion“;
    - Sei B_i die dabei neu erzeugte Variable;
- **für** $i = n - 1$ *bis* 1 **tue**
  - **für alle** $A_i \to A_j u \in P$, $j > i$ **tue**
    - Seien $A_j \to w_1 \mid \cdots \mid w_m$ alle Regeln mit A_j als linker Seite;
    - Ersetze $A_i \to A_j u$ durch $A_i \to w_1 u \mid \cdots \mid w_m u$ in P;
- **für** $i = 1$ *bis* n **tue**
  - **für alle** $B_i \to A_j u \in P$ **tue**
    - Seien $A_j \to w_1 \mid \cdots \mid w_m$ alle Regeln mit A_j als linker Seite;
    - Ersetze $B_i \to A_j u$ durch $B_i \to w_1 u \mid \cdots \mid w_m u$ in P;

Algorithmus 7: Herstellen der Greibach-Normalform

Eingabe: CFG $G = (\{A_1, \ldots, A_n\}, \Sigma, P, A_i)$ in Chomsky-NF mit $\varepsilon \notin L(G)$
Ausgabe: CFG G' in Greibach-Normalform mit $L(G) = L(G')$
Beginn

Ziel der geschachtelten Für-Schleife: Es gibt $A_i \to A_j u$ nur für $j > i$

- **für** $i = 1$ *bis* n **tue**
  - **für** $j = 1$ *bis* $i - 1$ **tue**
    - **für alle** $A_i \to A_j u \in P$ **tue**
      - Seien $A_j \to w_1 \mid \cdots \mid w_m$ alle Regeln in P mit A_j als linker Seite;
      - Ersetze $A_i \to A_j u$ durch $A_i \to w_1 u \mid \ldots \mid w_m u$ in P;

Ersetzen der Regeln $A_i \to A_j u$ mit $i > j$

  - **wenn** $A_i \to A_i u \in P$ **dann**
    - Eliminiere die Regel mit der Operation „Elimination der Links-Rekursion“;
    - Sei B_i die dabei neu erzeugte Variable;
- **für** $i = n - 1$ *bis* 1 **tue**
  - **für alle** $A_i \to A_j u \in P$, $j > i$ **tue**
    - Seien $A_j \to w_1 \mid \cdots \mid w_m$ alle Regeln mit A_j als linker Seite;
    - Ersetze $A_i \to A_j u$ durch $A_i \to w_1 u \mid \cdots \mid w_m u$ in P;
- **für** $i = 1$ *bis* n **tue**
  - **für alle** $B_i \to A_j u \in P$ **tue**
    - Seien $A_j \to w_1 \mid \cdots \mid w_m$ alle Regeln mit A_j als linker Seite;
    - Ersetze $B_i \to A_j u$ durch $B_i \to w_1 u \mid \cdots \mid w_m u$ in P;

Algorithmus 7: Herstellen der Greibach-Normalform

Eingabe: CFG $G = (\{A_1, \ldots, A_n\}, \Sigma, P, A_i)$ in Chomsky-NF mit $\varepsilon \notin L(G)$
Ausgabe: CFG G' in Greibach-Normalform mit $L(G) = L(G')$
Beginn

- **für** $i = 1$ *bis* n **tue**
  - **für** $j = 1$ *bis* $i - 1$ **tue**
    - **für alle** $A_i \to A_j u \in P$ **tue**
      - Seien $A_j \to w_1 \mid \cdots \mid w_m$ alle Regeln in P mit A_j als linker Seite;
      - Ersetze $A_i \to A_j u$ durch $A_i \to w_1 u \mid \ldots \mid w_m u$ in P;
  - **wenn** $A_i \to A_i u \in P$ **dann**
    - Eliminiere die Regel mit der Operation „Elimination der Links-Rekursion“;
    - Sei B_i die dabei neu erzeugte Variable;
- **für** $i = n - 1$ *bis* 1 **tue**
  - **für alle** $A_i \to A_j u \in P$, $j > i$ **tue**
    - Seien $A_j \to w_1 \mid \cdots \mid w_m$ alle Regeln mit A_j als linker Seite;
    - Ersetze $A_i \to A_j u$ durch $A_i \to w_1 u \mid \cdots \mid w_m u$ in P;
- **für** $i = 1$ *bis* n **tue**
  - **für alle** $B_i \to A_j u \in P$ **tue**
    - Seien $A_j \to w_1 \mid \cdots \mid w_m$ alle Regeln mit A_j als linker Seite;
    - Ersetze $B_i \to A_j u$ durch $B_i \to w_1 u \mid \cdots \mid w_m u$ in P;

Ziel der geschachtelten Für-Schleife: Es gibt $A_i \to A_j u$ nur für $j > i$

Ersetzen der Regeln $A_i \to A_j u$ mit $i > j$

Ersetzen der Regeln $A_i \to A_i u$

Algorithmus 7: Herstellen der Greibach-Normalform

Eingabe: CFG $G = (\{A_1, \ldots, A_n\}, \Sigma, P, A_i)$ in Chomsky-NF mit $\varepsilon \notin L(G)$
Ausgabe: CFG G' in Greibach-Normalform mit $L(G) = L(G')$
Beginn

für $i = 1$ *bis* n **tue**

Ziel der geschachtelten Für-Schleife: Es gibt $A_i \to A_j u$ nur für $j > i$

für $j = 1$ *bis* $i - 1$ **tue**

für alle $A_i \to A_j u \in P$ **tue**

Ersetzen der Regeln $A_i \to A_j u$ mit $i > j$

Seien $A_j \to w_1 \mid \cdots \mid w_m$ alle Regeln in P mit A_j als linker Seite;
Ersetze $A_i \to A_j u$ durch $A_i \to w_1 u \mid \ldots \mid w_m u$ in P;

wenn $A_i \to A_i u \in P$ **dann**

Ersetzen der Regeln $A_i \to A_i u$

Eliminiere die Regel mit der Operation „Elimination der Links-Rekursion“;
Sei B_i die dabei neu erzeugte Variable;

für $i = n - 1$ *bis* 1 **tue**

Nun gilt für $A_i \to A_j u$ stets $j > i$.
Damit gilt für $A_n \to u$: u beginnt mit Zeichen aus Σ.
Nächste Schleife: Ersetze alle $A_i \to A_j u$ mit $j > i$

für alle $A_i \to A_j u \in P$, $j > i$ **tue**

Seien $A_j \to w_1 \mid \cdots \mid w_m$ alle Regeln mit A_j als linker Seite;
Ersetze $A_i \to A_j u$ durch $A_i \to w_1 u \mid \cdots \mid w_m u$ in P;

für $i = 1$ *bis* n **tue**

für alle $B_i \to A_j u \in P$ **tue**

Seien $A_j \to w_1 \mid \cdots \mid w_m$ alle Regeln mit A_j als linker Seite;
Ersetze $B_i \to A_j u$ durch $B_i \to w_1 u \mid \cdots \mid w_m u$ in P;

Algorithmus 7: Herstellen der Greibach-Normalform

Eingabe: CFG $G = (\{A_1, \ldots, A_n\}, \Sigma, P, A_i)$ in Chomsky-NF mit $\varepsilon \notin L(G)$
Ausgabe: CFG G' in Greibach-Normalform mit $L(G) = L(G')$
Beginn

für $i = 1$ *bis* n **tue**

Ziel der geschachtelten Für-Schleife: Es gibt $A_i \to A_j u$ nur für $j > i$

für $j = 1$ *bis* $i - 1$ **tue**

für alle $A_i \to A_j u \in P$ **tue**

Ersetzen der Regeln $A_i \to A_j u$ mit $i > j$

Seien $A_j \to w_1 \mid \cdots \mid w_m$ alle Regeln in P mit A_j als linker Seite;
Ersetze $A_i \to A_j u$ durch $A_i \to w_1 u \mid \ldots \mid w_m u$ in P;

wenn $A_i \to A_i u \in P$ **dann**

Ersetzen der Regeln $A_i \to A_i u$

Eliminiere die Regel mit der Operation „Elimination der Links-Rekursion“;
Sei B_i die dabei neu erzeugte Variable;

für $i = n - 1$ *bis* 1 **tue**

Nun gilt für $A_i \to A_j u$ stets $j > i$.
Damit gilt für $A_n \to u$: u beginnt mit Zeichen aus Σ.
Nächste Schleife: Ersetze alle $A_i \to A_j u$ mit $j > i$

für alle $A_i \to A_j u \in P$, $j > i$ **tue**

Seien $A_j \to w_1 \mid \cdots \mid w_m$ alle Regeln mit A_j als linker Seite;
Ersetze $A_i \to A_j u$ durch $A_i \to w_1 u \mid \cdots \mid w_m u$ in P;

$w_1, \ldots, w_m$ fangen mit Zeichen aus Σ an, da Schleife absteigend läuft!

für $i = 1$ *bis* n **tue**

für alle $B_i \to A_j u \in P$ **tue**

Seien $A_j \to w_1 \mid \cdots \mid w_m$ alle Regeln mit A_j als linker Seite;
Ersetze $B_i \to A_j u$ durch $B_i \to w_1 u \mid \cdots \mid w_m u$ in P;

Algorithmus 7: Herstellen der Greibach-Normalform

Eingabe: CFG $G = (\{A_1, \ldots, A_n\}, \Sigma, P, A_i)$ in Chomsky-NF mit $\varepsilon \notin L(G)$
Ausgabe: CFG G' in Greibach-Normalform mit $L(G) = L(G')$
Beginn

für $i = 1$ *bis* n **tue**

Ziel der geschachtelten Für-Schleife: Es gibt $A_i \to A_j u$ nur für $j > i$

für $j = 1$ *bis* $i - 1$ **tue**

für alle $A_i \to A_j u \in P$ **tue**

Ersetzen der Regeln $A_i \to A_j u$ mit $i > j$

Seien $A_j \to w_1 \mid \cdots \mid w_m$ alle Regeln in P mit A_j als linker Seite;
Ersetze $A_i \to A_j u$ durch $A_i \to w_1 u \mid \ldots \mid w_m u$ in P;

wenn $A_i \to A_i u \in P$ **dann**

Ersetzen der Regeln $A_i \to A_i u$

Eliminiere die Regel mit der Operation „Elimination der Links-Rekursion“;
Sei B_i die dabei neu erzeugte Variable;

für $i = n - 1$ *bis* 1 **tue**

Nun gilt für $A_i \to A_j u$ stets $j > i$.
Damit gilt für $A_n \to u$: u beginnt mit Zeichen aus Σ.
Nächste Schleife: Ersetze alle $A_i \to A_j u$ mit $j > i$

für alle $A_i \to A_j u \in P$, $j > i$ **tue**

Seien $A_j \to w_1 \mid \cdots \mid w_m$ alle Regeln mit A_j als linker Seite;
Ersetze $A_i \to A_j u$ durch $A_i \to w_1 u \mid \cdots \mid w_m u$ in P;

$w_1, \ldots, w_m$ fangen mit Zeichen aus Σ an, da Schleife absteigend läuft!

für $i = 1$ *bis* n **tue**

Behandle die neuen Regeln mit B_i als linker Seite.

für alle $B_i \to A_j u \in P$ **tue**

Seien $A_j \to w_1 \mid \cdots \mid w_m$ alle Regeln mit A_j als linker Seite;
Ersetze $B_i \to A_j u$ durch $B_i \to w_1 u \mid \cdots \mid w_m u$ in P;

Korrektheit

Satz

Zu jeder CFG G mit $\varepsilon \notin L(G)$ gibt es eine CFG G' in Greibach-Normalform, sodass $L(G) = L(G')$ gilt.

Korrektheit folgt:

- durch Prüfen der genannten Invarianten
- Korrektheit der Elimination von Links-Rekursion
- Korrektheit der Operation „Inlining von Produktionen“

Ein Beispiel zur Herstellung der Greibach-Normalform ist im Skript

Widerlegen der Kontextfreiheit

Wir lernen Methoden kennen zum Widerlegen der Kontextfreiheit:

- Pumping-Lemma für kontextfreie Sprachen
- evtl. Ogdens-Lemma (evtl. in der Zentralübung)

Einschub: Binärbäume

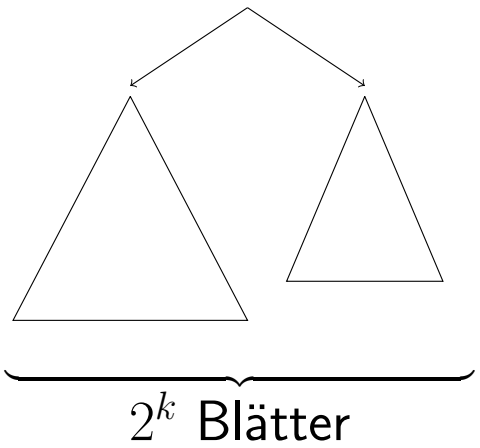
Binärbaum: Baum, wobei jeder Knoten 0 oder 2 Kinder hat

Lemma

Sei B ein Binärbaum mit $\geq 2^k$ Blättern. Dann hat B einen Pfad der Länge $\geq k$.

Beweis durch Induktion über k:

- $k = 0$: Ein Baum mit $2^k = 2^0 = 1$ Blättern besteht genau aus diesem Blatt und hat einen Pfad der Länge ≥ 0.
- $k > 0$: Einer der beiden Teilbäume unter der Wurzel hat $\geq 2^{k-1}$ Blätter.
- Per Induktionsannahme hat dieser einen Pfad der Länge $\geq k - 1$.
- Daher hat der gesamte Baum einen Pfad der Länge $\geq k$.



Pumping-Lemma für CFLs: Bemerkungen

- **Erinnerung:** Pumping-Lemma für reguläre Sprachen:
  Jede reguläre Sprache erfüllt die Pumping-Eigenschaft.
- Analog: Pumping-Lemma für kontextfreie Sprachen:
  Jede kontextfreie Sprache erfüllt die Pumping-Eigenschaft für kontextfreie Sprachen
- Kann vorallem zum Widerlegen benutzt werden:
  Sprache **verletzt** die Pumping-Eigenschaft für CFLs
  $\Longrightarrow$ Sprache ist **nicht kontextfrei**
- Pumping-Eigenschaft bei regulären Sprachen, informell:
  Man kann Worte an einer Stelle aufpumpen und verbleibt in der Sprache ($uv^iw \in L$ *für alle* $i \in \mathbb{N}$)
- Pumping-Eigenschaft bei kontextfreien Sprachen, informell:
  Man kann Worte an **zwei** *Stellen gleichzeitig aufpumpen und verbleibt in der Sprache* ($uv^iwx^iy \in L$ *für alle* $i \in \mathbb{N}$)

Pumping-Lemma für CFLs: Bemerkungen

- **Erinnerung:** Pumping-Lemma für reguläre Sprachen:
  Jede reguläre Sprache erfüllt die Pumping-Eigenschaft.
- Analog: Pumping-Lemma für kontextfreie Sprachen:
  Jede kontextfreie Sprache erfüllt die Pumping-Eigenschaft für kontextfreie Sprachen
- Kann vorallem zum Widerlegen benutzt werden:
  Sprache **verletzt** die Pumping-Eigenschaft für CFLs
  $\Longrightarrow$ Sprache ist **nicht kontextfrei**
- Pumping-Eigenschaft bei regulären Sprachen, informell:
  Man kann Worte an einer Stelle aufpumpen und verbleibt in der Sprache ($uv^iw \in L$ *für alle* $i \in \mathbb{N}$)
- Pumping-Eigenschaft bei kontextfreien Sprachen, informell:
  Man kann Worte an **zwei** *Stellen gleichzeitig aufpumpen und verbleibt in der Sprache* ($uv^iwx^iy \in L$ *für alle* $i \in \mathbb{N}$)

Pumping-Lemma für CFLs: Bemerkungen

- **Erinnerung:** Pumping-Lemma für reguläre Sprachen:
  Jede reguläre Sprache erfüllt die Pumping-Eigenschaft.
- Analog: Pumping-Lemma für kontextfreie Sprachen:
  Jede kontextfreie Sprache erfüllt die Pumping-Eigenschaft für kontextfreie Sprachen
- Kann vorallem zum Widerlegen benutzt werden:
  Sprache **verletzt** die Pumping-Eigenschaft für CFLs
  $\Longrightarrow$ Sprache ist **nicht kontextfrei**
- Pumping-Eigenschaft bei regulären Sprachen, informell:
  Man kann Worte an einer Stelle aufpumpen und verbleibt in der Sprache ($uv^iw \in L$ *für alle* $i \in \mathbb{N}$)
- Pumping-Eigenschaft bei kontextfreien Sprachen, informell:
  Man kann Worte an **zwei** *Stellen gleichzeitig aufpumpen und verbleibt in der Sprache* ($uv^iwx^iy \in L$ *für alle* $i \in \mathbb{N}$)

Pumping-Lemma für CFLs: Bemerkungen

- **Erinnerung:** Pumping-Lemma für reguläre Sprachen:
  Jede reguläre Sprache erfüllt die Pumping-Eigenschaft.
- Analog: Pumping-Lemma für kontextfreie Sprachen:
  Jede kontextfreie Sprache erfüllt die Pumping-Eigenschaft für kontextfreie Sprachen
- Kann vorallem zum Widerlegen benutzt werden:
  Sprache **verletzt** die Pumping-Eigenschaft für CFLs
  $\Longrightarrow$ Sprache ist **nicht kontextfrei**
- Pumping-Eigenschaft bei regulären Sprachen, informell:
  Man kann Worte an einer Stelle aufpumpen und verbleibt in der Sprache ($uv^iw \in L$ *für alle* $i \in \mathbb{N}$)
- Pumping-Eigenschaft bei kontextfreien Sprachen, informell:
  Man kann Worte an **zwei** *Stellen gleichzeitig aufpumpen und verbleibt in der Sprache* ($uv^iwx^iy \in L$ *für alle* $i \in \mathbb{N}$)

Pumping-Lemma für CFLs: Bemerkungen

- **Erinnerung:** Pumping-Lemma für reguläre Sprachen:
  Jede reguläre Sprache erfüllt die Pumping-Eigenschaft.
- Analog: Pumping-Lemma für kontextfreie Sprachen:
  Jede kontextfreie Sprache erfüllt die Pumping-Eigenschaft für kontextfreie Sprachen
- Kann vorallem zum Widerlegen benutzt werden:
  Sprache **verletzt** die Pumping-Eigenschaft für CFLs
  $\Longrightarrow$ Sprache ist **nicht kontextfrei**
- Pumping-Eigenschaft bei regulären Sprachen, informell:
  Man kann Worte an einer Stelle aufpumpen und verbleibt in der Sprache ($uv^iw \in L$ *für alle* $i \in \mathbb{N}$)
- Pumping-Eigenschaft bei kontextfreien Sprachen, informell:
  Man kann Worte an **zwei** *Stellen gleichzeitig aufpumpen und verbleibt in der Sprache* ($uv^iwx^iy \in L$ *für alle* $i \in \mathbb{N}$)

Das Pumping-Lemma für CFLs

Lemma (Pumping-Lemma für CFLs)

Sei L eine kontextfreie Sprache. Dann gibt es eine Zahl $n \in \mathbb{N}_{>0}$, sodass jedes Wort $z \in L$, das Mindestlänge n hat (d. h. $|z| \geq n$), als $z = uvwxy$ geschrieben werden kann, so dass gilt:

- $|vx| \geq 1$
- $|vwx| \leq n$
- für alle $i \geq 0$: $uv^iwx^iy \in L$.

Beweis des Pumping-Lemmas (1)

Beh.: Für jede CFL L gibt es $n \in \mathbb{N}_{>0}$, sodass jedes Wort $z \in L$ mit $|z| \geq n$ als $z = uvwxy$ geschrieben werden kann mit

- $|vx| \geq 1$
- $|vwx| \leq n$
- für alle $i \geq 0$: $uv^i wx^i y \in L$

Beweis:

- Sei $G = (V, \Sigma, P, S)$ eine CFG in Chomsky-NF mit $L(G) = L \setminus \{\varepsilon\}$

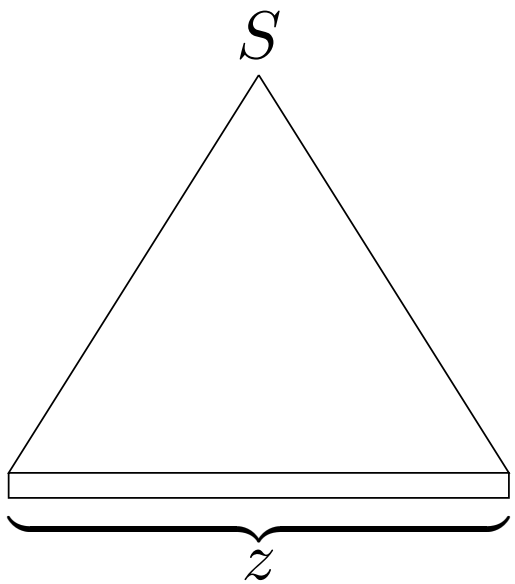
Beweis des Pumping-Lemmas (1)

Beh.: Für jede CFL L gibt es $n \in \mathbb{N}_{>0}$, sodass jedes Wort $z \in L$ mit $|z| \geq n$ als $z = uvwxy$ geschrieben werden kann mit

- $|vx| \geq 1$
- $|vwx| \leq n$
- für alle $i \geq 0$: $uv^iwx^iy \in L$

Beweis:

- Sei $G = (V, \Sigma, P, S)$ eine CFG in Chomsky-NF mit $L(G) = L \setminus \{\varepsilon\}$
- Betrachte Ableitung und Syntaxbaum eines Wortes z mit $|z| \geq 2^{|V|} = n$



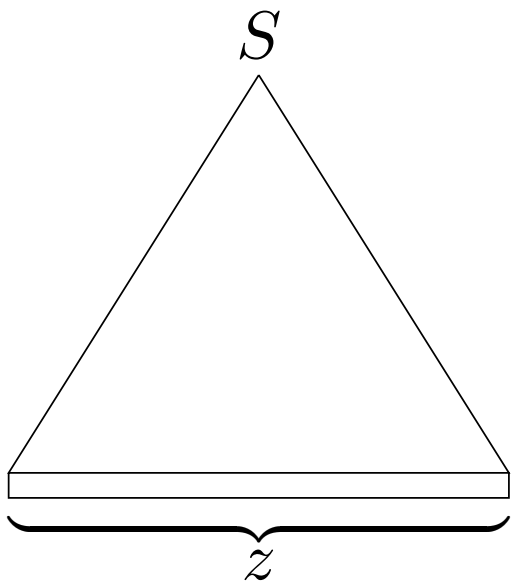
Beweis des Pumping-Lemmas (1)

Beh.: Für jede CFL L gibt es $n \in \mathbb{N}_{>0}$, sodass jedes Wort $z \in L$ mit $|z| \geq n$ als $z = uvwxy$ geschrieben werden kann mit

- $|vx| \geq 1$
- $|vwx| \leq n$
- für alle $i \geq 0$: $uv^iwx^iy \in L$

Beweis:

- Sei $G = (V, \Sigma, P, S)$ eine CFG in Chomsky-NF mit $L(G) = L \setminus \{\varepsilon\}$
- Betrachte Ableitung und Syntaxbaum eines Wortes z mit $|z| \geq 2^{|V|} = n$ (wenn es keine solche Ableitung gibt, gilt das Pumping-Lemma: es gibt dann keine Worte $z \in L$ mit Mindestlänge n)



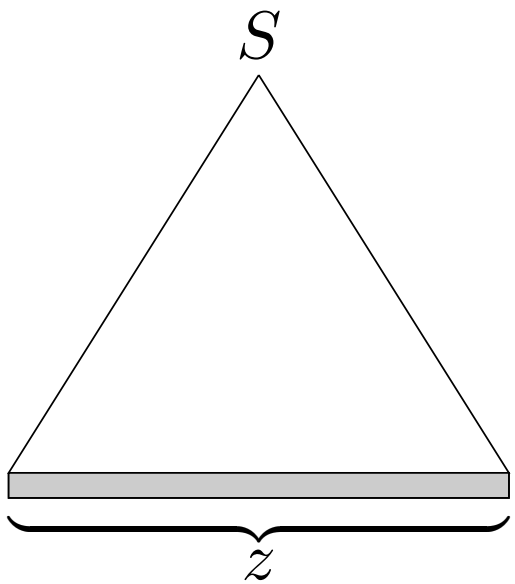
Beweis des Pumping-Lemmas (1)

Beh.: Für jede CFL L gibt es $n \in \mathbb{N}_{>0}$, sodass jedes Wort $z \in L$ mit $|z| \geq n$ als $z = uvwxy$ geschrieben werden kann mit

- $|vx| \geq 1$
- $|vwx| \leq n$
- für alle $i \geq 0$: $uv^iwx^iy \in L$

Beweis:

- Sei $G = (V, \Sigma, P, S)$ eine CFG in Chomsky-NF mit $L(G) = L \setminus \{\varepsilon\}$
- Betrachte Ableitung und Syntaxbaum eines Wortes z mit $|z| \geq 2^{|V|} = n$ (wenn es keine solche Ableitung gibt, gilt das Pumping-Lemma: es gibt dann keine Worte $z \in L$ mit Mindestlänge n)
- Da G in Chomsky-NF, ist der Syntaxbaum ein binärer Baum, bis auf die letzte Schicht, die Produktionen $A \to a$ anwendet



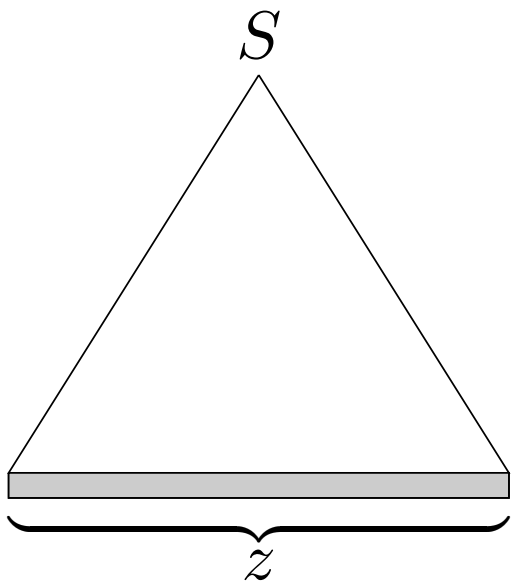
Beweis des Pumping-Lemmas (1)

Beh.: Für jede CFL L gibt es $n \in \mathbb{N}_{>0}$, sodass jedes Wort $z \in L$ mit $|z| \geq n$ als $z = uvwxy$ geschrieben werden kann mit

- $|vx| \geq 1$
- $|vwx| \leq n$
- für alle $i \geq 0$: $uv^iwx^iy \in L$

Beweis:

- Sei $G = (V, \Sigma, P, S)$ eine CFG in Chomsky-NF mit $L(G) = L \setminus \{\varepsilon\}$
- Betrachte Ableitung und Syntaxbaum eines Wortes z mit $|z| \geq 2^{|V|} = n$ (wenn es keine solche Ableitung gibt, gilt das Pumping-Lemma: es gibt dann keine Worte $z \in L$ mit Mindestlänge n)
- Da G in Chomsky-NF, ist der Syntaxbaum ein binärer Baum, bis auf die letzte Schicht, die Produktionen $A \to a$ anwendet
- Baum ohne letzte Schicht hat $|z| \geq 2^{|V|}$ Blätter.



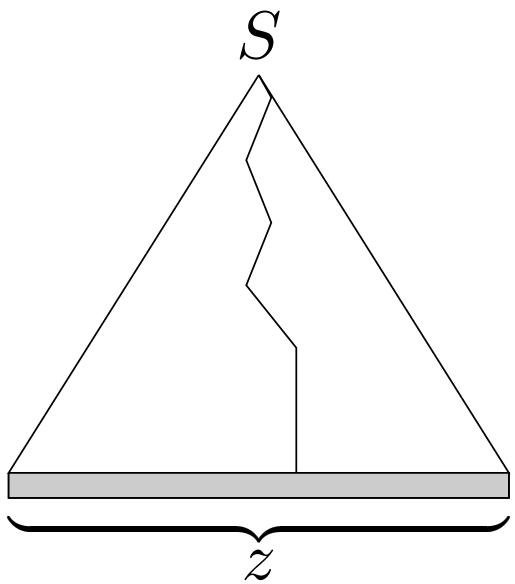
Beweis des Pumping-Lemmas (1)

Beh.: Für jede CFL L gibt es $n \in \mathbb{N}_{>0}$, sodass jedes Wort $z \in L$ mit $|z| \geq n$ als $z = uvwxy$ geschrieben werden kann mit

- $|vx| \geq 1$
- $|vwx| \leq n$
- für alle $i \geq 0$: $uv^iwx^iy \in L$

Beweis:

- Sei $G = (V, \Sigma, P, S)$ eine CFG in Chomsky-NF mit $L(G) = L \setminus \{\varepsilon\}$
- Betrachte Ableitung und Syntaxbaum eines Wortes z mit $|z| \geq 2^{|V|} = n$ (wenn es keine solche Ableitung gibt, gilt das Pumping-Lemma: es gibt dann keine Worte $z \in L$ mit Mindestlänge n)
- Da G in Chomsky-NF, ist der Syntaxbaum ein binärer Baum, bis auf die letzte Schicht, die Produktionen $A \to a$ anwendet
- Baum ohne letzte Schicht hat $|z| \geq 2^{|V|}$ Blätter.
- Daher gibt es einen Pfad von der Wurzel zum Blatt, der Länge $\geq |V|$, der aus $\geq |V + 1|$ Knoten besteht und jeder Knoten ist mit einer Variablen markiert.



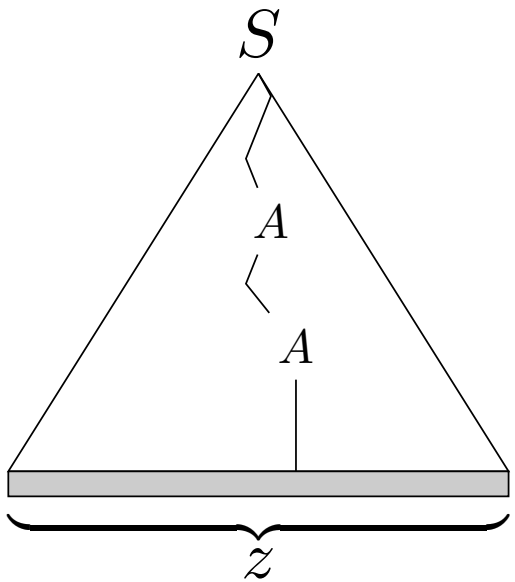
Beweis des Pumping-Lemmas (2)

Beh.: Für jede CFL L gibt es $n \in \mathbb{N}_{>0}$, sodass jedes Wort $z \in L$ mit $|z| \geq n$ als $z = uvwxy$ geschrieben werden kann mit

- $|vx| \geq 1$
- $|vwx| \leq n$
- für alle $i \geq 0$: $uv^iwx^iy \in L$

Beweis:

- ...
- Da es nur $|V|$ Variablen gibt, kommt mindestens eine Variable mehrfach auf diesem Pfad vor.
- Wähle die Vorkommen der Variablen so, dass das zweite Vorkommen von unten gesehen am tiefsten ist. Sei A die Variable.



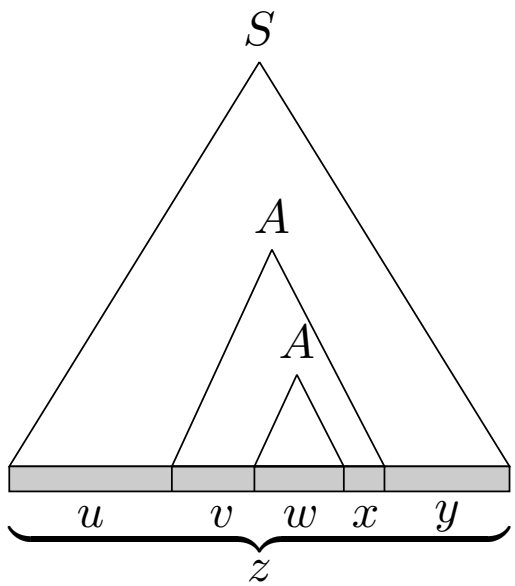
Beweis des Pumping-Lemmas (3)

Beh.: Für jede CFL L gibt es $n \in \mathbb{N}_{>0}$, sodass jedes Wort $z \in L$ mit $|z| \geq n$ als $z = uvwxy$ geschrieben werden kann mit

- $|vx| \geq 1$
- $|vwx| \leq n$
- für alle $i \geq 0$: $uv^iwx^iy \in L$

Beweis:

- Betrachte die Teilbäume, die jeweils A als Wurzel haben.
- Sie entsprechen Ableitungen von Teilworten von z
- Der Teilbaum mit dem unteren A als Wurzel erzeugt ein Teilwort des Teilbaums mit dem oberen A als Wurzel. D.h. $z = uvwxy$, wobei vwx vom oberen A und w vom unteren A erzeugt wird.
- Es gilt $|w| \geq 1$, da Variablen einer Grammatik in Chomsky-NF nur Wörter mit Länge ≥ 1 herleiten
- Das Wort vwx muss echt länger sein als w, da das obere A über dem unteren A steht. Daher folgt $|v| \geq 1$ und/oder $|x| \geq 1$ und somit $|vx| \geq 1$.



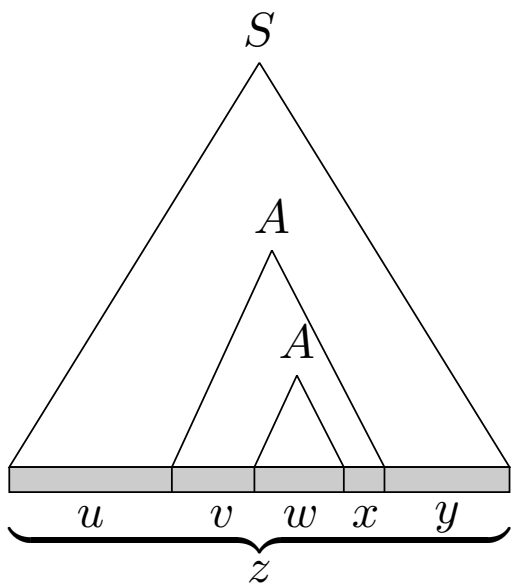
Beweis des Pumping-Lemmas (4)

Beh.: Für jede CFL L gibt es $n \in \mathbb{N}_{>0}$, sodass jedes Wort $z \in L$ mit $|z| \geq n$ als $z = uvwxy$ geschrieben werden kann mit

- $|vx| \geq 1$
- $|vwx| \leq n$
- für alle $i \geq 0$: $uv^iwx^iy \in L$

Beweis:

- Da wir das tiefste Vorkommen der wiederholten Variable gewählt haben, kann der Pfad vom oberen A bis zur Blattebene nur aus $\leq |V| + 1$ Knoten bestehen und Länge $\leq |V|$ haben
- Daraus folgt: $|vwx| \leq 2^{|V|} = n$
- Aus dem Baum folgt: $A \Rightarrow^* w$ und $A \Rightarrow^* vAx$ und daher kann man auch $A \Rightarrow^* v^iwx^i$ für alle $i \in \mathbb{N}$ ableiten
- Schließlich folgt daraus $S \Rightarrow^* uv^iwx^iy$ für alle $i \in \mathbb{N}$.



Pumping-Lemma: Illustrationen

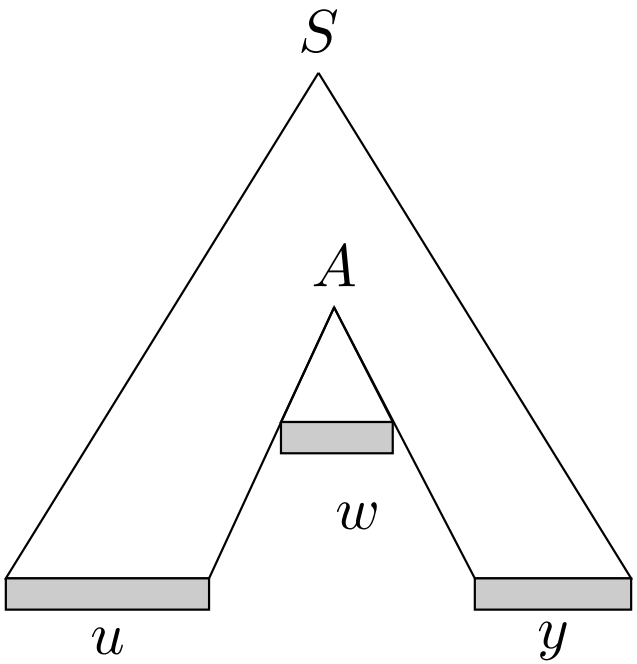


Illustration für uv^0wx^0y

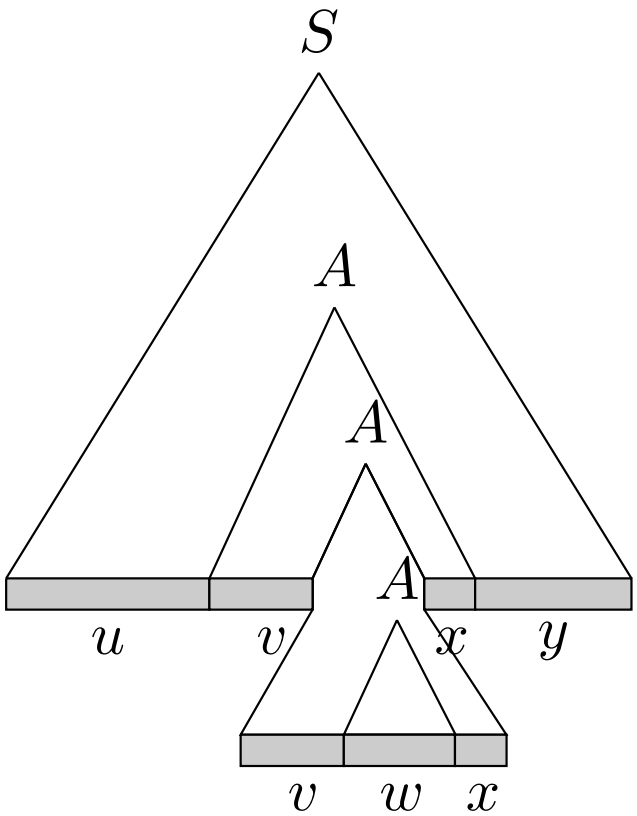


Illustration für uv^2wx^2y

Verwendung des Pumping-Lemma

- Die Pumping-Eigenschaft ist eine notwendige aber **keine hinreichende** Bedingung für CFLs.
- Daher kann das Pumping-Lemma **nicht** verwendet werden, um Kontextfreiheit zu zeigen
- Aber: Es kann verwendet werden, um **Kontextfreiheit zu widerlegen**

Pumping-Lemma zum Widerlegen der Kontextfreiheit

Formulierung des Pumping-Lemmas für CFGs zum Widerlegen der Kontextfreiheit

Sei L eine formale Sprache für die gilt: Für jede Zahl $n \in \mathbb{N}_{>0}$ gibt es ein $z \in L$, das Mindestlänge n hat (d. h. $|z| \geq n$), und für jede Zerlegung $z = uvwxy$ mit $|vwx| \leq n$ und $|vx| \geq 1$, gibt es ein $i \geq 0$, sodass $uv^iwx^iy \notin L$. Dann ist L nicht kontextfrei.

Beweis: Umformung der negierten prädikatenlogischen Formel (siehe Skript), die sich aus dem Pumping-Lemma ergibt.

Pumping-Lemma als Spiel

Sei L die formale Sprache.

1. Der **Gegner** wählt die Zahl $n \in \mathbb{N}_{>0}$.
2. **Wir** wählen das Wort $z \in L$ mit $|z| \geq n$.
3. Der **Gegner** wählt die Zerlegung $z = uvwxy$ mit $|vx| \geq 1$ und $|vwx| \leq n$
4. **Wir** gewinnen das Spiel, wenn wir ein $i \geq 0$ angeben können, sodass $uv^iwx^iw \notin L$.

Wenn wir **für jede Wahl des Gegners** das Spiel gewinnen können, dann haben wir gezeigt, dass L nicht kontextfrei ist.

Beispiele

Satz

Die Sprache $L = \{a^l b^l c^l \mid l \in \mathbb{N}\}$ ist nicht kontextfrei.

Beweis:

- Gegner wählt $n \in \mathbb{N}_{>0}$
- Wir wählen $z = a^n b^n c^n$.
- Gegner wählt Zerlegung $z = uvwxy$ mit $|vx| \geq 1$ und $|vwx| \leq n$
- Fall 1: vwx ist von der Form $a^i b^j$, $i + j \leq n$
  Da $|vx| \geq 1$, gilt $\#_a(vx) \geq 1$ oder $\#_b(vx) \geq 1$, aber $\#_c(vx) = 0$
  Damit folgt $uv^0wx^0y \notin L$
- Fall 2: vwx ist von der Form $b^i c^j$, $i + j \leq n$
  Da $|vx| \geq 1$, gilt $\#_b(vx) \geq 1$ oder $\#_c(vx) \geq 1$, aber $\#_a(vx) = 0$
  Damit folgt $uv^0wx^0y \notin L$
- Andere Fälle sind nicht möglich!

Beispiele

Satz

Die Sprache $L = \{a^l b^l c^l \mid l \in \mathbb{N}\}$ ist nicht kontextfrei.

Beweis:

- Gegner wählt $n \in \mathbb{N}_{>0}$
- Wir wählen $z = a^n b^n c^n$.
- Gegner wählt Zerlegung $z = uvwxy$ mit $|vx| \geq 1$ und $|vwx| \leq n$
- Fall 1: vwx ist von der Form $a^i b^j$, $i + j \leq n$
  Da $|vx| \geq 1$, gilt $\#_a(vx) \geq 1$ oder $\#_b(vx) \geq 1$, aber $\#_c(vx) = 0$
  Damit folgt $uv^0wx^0y \notin L$
- Fall 2: vwx ist von der Form $b^i c^j$, $i + j \leq n$
  Da $|vx| \geq 1$, gilt $\#_b(vx) \geq 1$ oder $\#_c(vx) \geq 1$, aber $\#_a(vx) = 0$
  Damit folgt $uv^0wx^0y \notin L$
- Andere Fälle sind nicht möglich!

Beispiele

Satz

Die Sprache $L = \{a^l b^l c^l \mid l \in \mathbb{N}\}$ ist nicht kontextfrei.

Beweis:

- Gegner wählt $n \in \mathbb{N}_{>0}$
- Wir wählen $z = a^n b^n c^n$.
- Gegner wählt Zerlegung $z = uvwxy$ mit $|vx| \geq 1$ und $|vwx| \leq n$
- Fall 1: vwx ist von der Form $a^i b^j$, $i + j \leq n$
  Da $|vx| \geq 1$, gilt $\#_a(vx) \geq 1$ oder $\#_b(vx) \geq 1$, aber $\#_c(vx) = 0$
  Damit folgt $uv^0wx^0y \notin L$
- Fall 2: vwx ist von der Form $b^i c^j$, $i + j \leq n$
  Da $|vx| \geq 1$, gilt $\#_b(vx) \geq 1$ oder $\#_c(vx) \geq 1$, aber $\#_a(vx) = 0$
  Damit folgt $uv^0wx^0y \notin L$
- Andere Fälle sind nicht möglich!

Beispiele

Satz

Die Sprache $L = \{a^l b^l c^l \mid l \in \mathbb{N}\}$ ist nicht kontextfrei.

Beweis:

- Gegner wählt $n \in \mathbb{N}_{>0}$
- Wir wählen $z = a^n b^n c^n$.
- Gegner wählt Zerlegung $z = uvwxy$ mit $|vx| \geq 1$ und $|vwx| \leq n$
- Fall 1: vwx ist von der Form $a^i b^j$, $i + j \leq n$
  Da $|vx| \geq 1$, gilt $\#_a(vx) \geq 1$ oder $\#_b(vx) \geq 1$, aber $\#_c(vx) = 0$
  Damit folgt $uv^0wx^0y \notin L$
- Fall 2: vwx ist von der Form $b^i c^j$, $i + j \leq n$
  Da $|vx| \geq 1$, gilt $\#_b(vx) \geq 1$ oder $\#_c(vx) \geq 1$, aber $\#_a(vx) = 0$
  Damit folgt $uv^0wx^0y \notin L$
- Andere Fälle sind nicht möglich!

□

Beispiele

Satz

Die Sprache $L = \{a^l b^l c^l \mid l \in \mathbb{N}\}$ ist nicht kontextfrei.

Beweis:

- Gegner wählt $n \in \mathbb{N}_{>0}$
- Wir wählen $z = a^n b^n c^n$.
- Gegner wählt Zerlegung $z = uvwxy$ mit $|vx| \geq 1$ und $|vwx| \leq n$
- Fall 1: vwx ist von der Form $a^i b^j$, $i + j \leq n$
  Da $|vx| \geq 1$, gilt $\#_a(vx) \geq 1$ oder $\#_b(vx) \geq 1$, aber $\#_c(vx) = 0$
  Damit folgt $uv^0wx^0y \notin L$
- Fall 2: vwx ist von der Form $b^i c^j$, $i + j \leq n$
  Da $|vx| \geq 1$, gilt $\#_b(vx) \geq 1$ oder $\#_c(vx) \geq 1$, aber $\#_a(vx) = 0$
  Damit folgt $uv^0wx^0y \notin L$
- Andere Fälle sind nicht möglich!

□

Beispiele

Satz

Die Sprache $L = \{a^l b^l c^l \mid l \in \mathbb{N}\}$ ist nicht kontextfrei.

Beweis:

- Gegner wählt $n \in \mathbb{N}_{>0}$
- Wir wählen $z = a^n b^n c^n$.
- Gegner wählt Zerlegung $z = uvwxy$ mit $|vx| \geq 1$ und $|vwx| \leq n$
- Fall 1: vwx ist von der Form $a^i b^j$, $i + j \leq n$
  Da $|vx| \geq 1$, gilt $\#_a(vx) \geq 1$ oder $\#_b(vx) \geq 1$, aber $\#_c(vx) = 0$
  Damit folgt $uv^0wx^0y \notin L$
- Fall 2: vwx ist von der Form $b^i c^j$, $i + j \leq n$
  Da $|vx| \geq 1$, gilt $\#_b(vx) \geq 1$ oder $\#_c(vx) \geq 1$, aber $\#_a(vx) = 0$
  Damit folgt $uv^0wx^0y \notin L$
- Andere Fälle sind nicht möglich!

Beispiele (2)

Satz

Die Sprache $L = \{a^i b^j c^i d^j \mid i, j \in \mathbb{N}_{>0}\}$ ist nicht kontextfrei.

Beweis:

- Gegner wählt $n \in \mathbb{N}_{>0}$.
- Wir wählen $z = a^n b^n c^n d^n$.
- Gegner wählt Zerlegung $z = uvwxy$ mit $|vx| \geq 1$ und $|vwx| \leq n$
- 1.Fall: $vwx = a^i b^j$ mit $i + j \leq n$. Da $|vx| \geq 1$, gilt $\#_a(vx) + \#_b(vx) \geq 1$ und $uv^0wx^0y = uwy = a^{i'} b^{j'} c^n d^n$ und $i' < n$ und/oder $j' < n$, d.h. $uwy \notin L$.
- 2.Fall: $vwx = b^i c^j$ mit $i + j \leq n$. Da $|vx| \geq 1$, gilt $\#_b(vx) + \#_c(vx) \geq 1$ und $uv^0wx^0y = uwy = a^n b^{i'} c^{j'} d^n$ und $i' < n$ und/oder $j' < n$, d.h. $uwy \notin L$
- 3.Fall: $vwx = c^i d^j$ mit $i + j \leq n$. Da $|vx| \geq 1$, gilt $\#_c(vx) + \#_d(vx) \geq 1$ und $uv^0wx^0y = uwy = a^n b^n c^{i'} d^{j'}$ und $i' < n$ und/oder $j' < n$, d.h. $uwy \notin L$.

Unäres Alphabet

Satz

Sei L eine formale Sprache über einem unären Alphabet (d.h. $|\Sigma| = 1$). Dann ist L genau dann regulär, wenn L kontextfrei ist.

Beweis:

- Wenn L regulär ist, dann ist L auch kontextfrei.
- Rückrichtung: Siehe Skript
  (Beweis verwendet die Pumping-Eigenschaft für CFLs und konstruiert daraus eine Vereinigung von regulären Sprachen)

Beispiele

Satz

Die Sprachen

$$\begin{aligned} L_1 &= \{a^p \mid p \text{ ist eine Primzahl}\} \\ L_2 &= \{a^n \mid n \text{ ist keine Primzahl}\} \\ L_3 &= \{a^n \mid n \text{ ist Quadratzahl}\} \\ L_4 &= \{a^{2^n} \mid n \in \mathbb{N}\} \end{aligned}$$

sind allesamt nicht kontextfrei.

Beweis: Wir haben für alle 4 Sprachen gezeigt, dass sie nicht regulär sind. Da sie alle über einem unären Alphabet definiert sind, sind sie auch nicht kontextfrei.

Abschlusseigenschaften der kontextfreien Sprachen

Theorem

Die kontextfreien Sprachen sind abgeschlossen unter Vereinigung, Produkt und Kleeneschem Abschluss.

Beweis:

- Seien L_1,L_2 CFLs und $G_i = (V_i, \Sigma_i, P_i, S_i)$ CFGs mit $L(G_i) = L_i$ für $i = 1, 2$. O.B.d.A. sei $V_1 \cap V_2 = \emptyset$.
- Seien S, S' neue Variablen ($\{S, S'\} \cap (V_1 \cup V_2) = \emptyset$).
- Vereinigung: Es gilt: $L(G_\cup) = L(G_1) \cup L(G_2) = L_1 \cup L_2$ für $G_\cup = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, P_1 \cup P_2 \cup \{S \to S_1 \mid S_2\}, S)$.
- Produkt: Es gilt $L(G_\circ) = L(G_1)L(G_2) = L_1 L_2$ für $G_\circ = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, P_1 \cup P_2 \cup \{S \to S_1 S_2\}, S)$.
- Kleenescher Abschluss: Sei $G_{1,*} = (V_1 \cup \{S', S_0\}, \Sigma, P', S')$ mit $P' = (P \setminus \{S_1 \to \varepsilon\}) \cup \{S' \to \varepsilon, S' \to S, S \to SS, S \to S_1\}$. Dann gilt $L(G_{1,*}) = L(G_1)^*$.

Abschlusseigenschaften der kontextfreien Sprachen

Theorem

Die kontextfreien Sprachen sind abgeschlossen unter Vereinigung, Produkt und Kleeneschem Abschluss.

Beweis:

- Seien L_1,L_2 CFLs und $G_i = (V_i, \Sigma_i, P_i, S_i)$ CFGs mit $L(G_i) = L_i$ für $i = 1, 2$. O.B.d.A. sei $V_1 \cap V_2 = \emptyset$.
- Seien S, S' neue Variablen ($\{S, S'\} \cap (V_1 \cup V_2) = \emptyset$).
- Vereinigung: Es gilt: $L(G_\cup) = L(G_1) \cup L(G_2) = L_1 \cup L_2$ für $G_\cup = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, P_1 \cup P_2 \cup \{S \to S_1 \mid S_2\}, S)$.
- Produkt: Es gilt $L(G_\circ) = L(G_1)L(G_2) = L_1 L_2$ für $G_\circ = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, P_1 \cup P_2 \cup \{S \to S_1 S_2\}, S)$.
- Kleenescher Abschluss: Sei $G_{1,*} = (V_1 \cup \{S', S_0\}, \Sigma, P', S')$ mit $P' = (P \setminus \{S_1 \to \varepsilon\}) \cup \{S' \to \varepsilon, S' \to S, S \to SS, S \to S_1\}$. Dann gilt $L(G_{1,*}) = L(G_1)^*$.

Abschlusseigenschaften der kontextfreien Sprachen

Theorem

Die kontextfreien Sprachen sind abgeschlossen unter Vereinigung, Produkt und Kleeneschem Abschluss.

Beweis:

- Seien L_1,L_2 CFLs und $G_i = (V_i, \Sigma_i, P_i, S_i)$ CFGs mit $L(G_i) = L_i$ für $i = 1, 2$. O.B.d.A. sei $V_1 \cap V_2 = \emptyset$.
- Seien S, S' neue Variablen ($\{S, S'\} \cap (V_1 \cup V_2) = \emptyset$).
- Vereinigung: Es gilt: $L(G_\cup) = L(G_1) \cup L(G_2) = L_1 \cup L_2$ für $G_\cup = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, P_1 \cup P_2 \cup \{S \to S_1 \mid S_2\}, S)$.
- Produkt: Es gilt $L(G_\circ) = L(G_1)L(G_2) = L_1 L_2$ für $G_\circ = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, P_1 \cup P_2 \cup \{S \to S_1 S_2\}, S)$.
- Kleenescher Abschluss: Sei $G_{1,*} = (V_1 \cup \{S', S_0\}, \Sigma, P', S')$ mit $P' = (P \setminus \{S_1 \to \varepsilon\}) \cup \{S' \to \varepsilon, S' \to S, S \to SS, S \to S_1\}$. Dann gilt $L(G_{1,*}) = L(G_1)^*$.

Abschlusseigenschaften der kontextfreien Sprachen

Theorem

Die kontextfreien Sprachen sind abgeschlossen unter Vereinigung, Produkt und Kleeneschem Abschluss.

Beweis:

- Seien L_1,L_2 CFLs und $G_i = (V_i, \Sigma_i, P_i, S_i)$ CFGs mit $L(G_i) = L_i$ für $i = 1, 2$. O.B.d.A. sei $V_1 \cap V_2 = \emptyset$.
- Seien S, S' neue Variablen ($\{S, S'\} \cap (V_1 \cup V_2) = \emptyset$).
- Vereinigung: Es gilt: $L(G_\cup) = L(G_1) \cup L(G_2) = L_1 \cup L_2$ für $G_\cup = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, P_1 \cup P_2 \cup \{S \to S_1 \mid S_2\}, S)$.
- Produkt: Es gilt $L(G_\circ) = L(G_1)L(G_2) = L_1 L_2$ für $G_\circ = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, P_1 \cup P_2 \cup \{S \to S_1 S_2\}, S)$.
- Kleenescher Abschluss: Sei $G_{1,*} = (V_1 \cup \{S', S_0\}, \Sigma, P', S')$ mit $P' = (P \setminus \{S_1 \to \varepsilon\}) \cup \{S' \to \varepsilon, S' \to S, S \to SS, S \to S_1\}$. Dann gilt $L(G_{1,*}) = L(G_1)^*$.

Abschlusseigenschaften der kontextfreien Sprachen (2)

Theorem

Die kontextfreien Sprachen sind **nicht abgeschlossen** unter Schnitt- und Komplementbildung.

- Sei $L_1 = \{a^n b^m c^m \mid m, n \in \mathbb{N}\}$ und sei $L_2 = \{a^m b^m c^n \mid m, n \in \mathbb{N}\}$
- $G_1 = (\{A, D, S\}, \{a, b, c\}, \{S \to AD, A \to \varepsilon \mid aA, D \to bDc \mid \varepsilon\}, S)$
  $G_2 = (\{C, D, S\}, \{a, b, c\}, \{S \to DC, C \to cC \mid \varepsilon, D \to aDb \mid \varepsilon\}, S)$
  Für i=1,2: $L(G_i) = L_i$, daher: L_1 und L_2 sind beide kontextfrei.
- $L_1 \cap L_2 = \{a^n b^n c^n \mid n \in \mathbb{N}_{>0}\}$ ist nicht kontextfrei (bereits gezeigt).
- Daher sind die CFLs nicht abgeschlossen bezüglich Schnittbildung.
- Komplement: Widerspruchsbeweis: Annahme: L CFL $\implies$ $\overline{L}$ CFL
- Seien L_1, L_2 CFLs. Dann ist auch $\overline{\overline{L_1} \cup \overline{L_2}}$ CFL
  (da CFLs abgeschlossen bez. $\bar{\cdot}$ und $\cup$)
- Aber: $\overline{\overline{L_1} \cup \overline{L_2}} = L_1 \cap L_2$. Widerspruch!

Zusammenfassung

- Normalformen für CFGs: Chomsky-Normalform und Greibach-Normalform
- Widerlegen der Kontextfreiheit mit dem Pumping-Lemma
- Abschlusseigenschaften kontextfreier Sprachen