Theoretische Informatik für Wirtschaftsinformatik und Lehramt

Eigenschaften kontextfreier Sprachen

Priv.-Doz. Dr. Stefan Milius
stefan.milius@fau.de

Theoretische Informatik
Friedrich-Alexander Universität Erlangen-Nürnberg

SS 2016

Gliederung

Worum geht es in diesem Abschnitt? (I)

- **Abschlusseigenschaften:** aus einfachen komplexere kontextfreie Sprachen konstruieren
- **Normalformen** kontextfreier Grammatiken (inbesondere **Chomsky-Normalform**):

  kontextfreie Grammatiken so umformen, dass die Regeln eine einfache Form haben, aber dieselbe Sprache erzeugt wird
- **Entscheidbarkeit** kontextfreier Sprachen: Algorithmus von Cocke-Younger-Kasami (CYK) für das Wortproblem
- **Pumping Lemma** für kontextfreie Sprachen: Methode, um nachzuweisen, dass eine Sprache nicht kontextfrei ist

Lernziele

- Abschlusseigenschaften kontextfreier Sprachen kennen und beweisen können
- eine kontextfreie Grammatik schrittweise normalisieren können
- den CYK-Algorithmus anwenden können, um zu prüfen, ob ein gegebenes Wort von einer Grammatik erzeugt wird
- Pumping-Lemma verwenden können, um zu beweisen, dass eine gegebene Sprache nicht kontextfrei ist.

Abschlusseigenschaften kontextfreier Sprachen

Abschlusseigenschaften kontextfreier Sprachen (I)

Satz 5.1 (Abschusseigenschaften)

Die kontextfreien Sprachen sind abgeschlossen unter den Operationen Vereinigung, Produkt, Potenz, Kleene-Stern und Spiegelung.

Sind L, L_1, L_2 kontextfreie Sprachen und $n \in \mathbb{N}$, so sind die folgenden Sprachen kontextfrei:

a) $L_1 \cup L_2$

b) $L_1 \circ L_2$

c) L^n

d) L^*

e) L^R

Abschlusseigenschaften kontextfreier Sprachen (II)

Beweis: Seien L_1, L_2 kontextfreie Sprachen.

a) $L_1 \cup L_2$ kontextfrei

Seien $G_1 = (V_1, \Sigma, P_1, S_1)$, $G_2 = (V_2, \Sigma, P_2, S_2)$ kfG mit

$$L(G_1) = L_1 \qquad \text{und} \qquad L(G_2) = L_2.$$

O.B.d.A.: $V_1 \cap V_2 = \emptyset$.

Nehme ein neues Startsymbol $S \notin V_1 \cup V_2 \cup \Sigma$ und konstruiere die kontextfreie Grammatik

$$G = (V_1 \cup V_2 \cup \{S\}, \Sigma, P_1 \cup P_2 \cup \{S \to S_1 \mid S_2\}, S)$$

Dann gilt: Ableitungen von $w \in \Sigma^*$ mit G haben die Form

$$S \Rightarrow S_1 \Rightarrow^* w \qquad \text{oder} \qquad S \Rightarrow S_2 \Rightarrow^* w$$

Also: $L(G) = L_1 \cup L_2$ ist kontextfrei.

Abschlusseigenschaften kontextfreier Sprachen (II)

Beweis: (Fortsetzung) Seien L_1, L_2 kontextfreie Sprachen.

b) $L_1 \circ L_2$ kontextfrei

Analog: forme

$$G = (V_1 \cup V_2 \cup \{S\}, \Sigma, P_1 \cup P_2 \cup \{S \to S_1 S_2\}, S)$$

Dann gilt: $L(G) = L_1 \circ L_2$ ist kontextfrei.

Abschlusseigenschaften kontextfreier Sprachen (III)

Beweis (Fortsetzung): Sei L kontextfreie Sprache.

c) L^n kontextfrei für jedes $n \in \mathbb{N}$:

$L^0 = \{\varepsilon\}$ und $L^1 = L$ sind kontextfrei.

Für $n \geq 2$: $L^n = L \circ L \circ L \circ \cdots \circ L$ kontextfrei
(n-mal b) anwenden)

Abschlusseigenschaften kontextfreier Sprachen (III)

Beweis (Fortsetzung): Sei L kontextfreie Sprache.

d) L^* kontextfrei:

Sei $G = (V, \Sigma, P, S)$ kontextfreie Grammatik mit $L(G) = L$.

Nehme neues Startsymbol $S' \notin V \cup \Sigma$ und konstruiere die kontextfreie Grammatik

$$G' = (V \cup \{S'\}, \Sigma, P \cup \{S' \to \varepsilon \mid S'S\}, S')$$

Man beweist leicht: $L(G') = L^*$.

Also ist L^* kontextfrei.

Abschlusseigenschaften kontextfreier Sprachen (IV)

Beweis (Fortsetzung): Sei L kontextfreie Sprache.

e) L^R kontextfrei

Sei $G = (V, \Sigma, P, S)$ kontextfreie Grammatik mit $L(G) = L$.

Kontruiere die kontextfreie Grammatik $G' = (V, \Sigma, P', S)$ mit

$$P' = \{ A \to w^R \mid (A \to w) \in P \}$$

Behauptung: Es gilt für alle $A \in V$ und $w \in (V \cup \Sigma)^*$:

$$A \Rightarrow_G^* w \quad \Longleftrightarrow \quad A \Rightarrow_{G'}^* w^R.$$

(Daraus folgt $L(G') = L^R$ ist kontextfrei.)

Abschlusseigenschaften kontextfreier Sprachen (IV)

Beweis (Fortsetzung): Sei L kontextfreie Sprache.

e) L^R kontextfrei

Beweis der Behauptung

$$A \Rightarrow_G^* w \quad \Longleftrightarrow \quad A \Rightarrow_{G'}^* w^R.$$

Per Induktion über die Länge n der Ableitung $A \Rightarrow_G^* w$:

1. $n = 0$: Dann ist $w = A$ und $w^R = A$, also „ $\Longleftrightarrow$ “ trivial
2. $n > 0$: Dann gilt (für eine Regel $B \to x$ in P)

$$\begin{aligned} A \Rightarrow_G^* w \quad &\Longleftrightarrow \quad A \Rightarrow_G^* uBv \Rightarrow_G uxv = w \\ &\Longleftrightarrow \quad A \Rightarrow_{G'}^* (uBv)^R = v^R B u^R \\ &\qquad\quad \Rightarrow_{G'} v^R x^R u^R = (uxv)^R = w^R \\ &\Longleftrightarrow \quad A \Rightarrow_{G'}^* w^R \end{aligned}$$

Abschlusseigenschaften kontextfreier Sprachen (V)

Bemerkung

Die kontextfreien Sprachen sind nicht unter Durchschnitt und Komplement abgeschlossen.

Denn:

- die folgenden Spachen sind kontextfrei:

  $$L_1 = \{a^k b^n c^n \mid k, n \in \mathbb{N}\} \quad \text{und} \quad L_2 = \{a^n b^n c^k \mid k, n \in \mathbb{N}\}$$

  Aber: $L_1 \cap L_2 = \{a^n b^n c^n \mid n \in \mathbb{N}\}$ ist nicht kontextfrei (Beweis später)

- Es gilt: Abschluss unter $\cup$ und $\overline{(-)} \implies$ Abschluss unter $\cap$

  Denn: $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$

  Also: kfS sind nicht unter $\overline{(-)}$ abgeschlossen sein.

Normalformen kontextfreier Grammatiken

Normalformen

- Allgemeine Regelform bei kontextfreien Grammatiken $G = (V, \Sigma, P, S)$:

$$A \to w \qquad A \in V, w \in (V \cup \Sigma)^*$$

- **Jetzt:** Einschränkung auf speziellere Formen ohne die „Mächtigkeit“ der Grammatik-Klasse einzuschränken.
- **Zweck:** Vereinfachung weiterer Algorithmen und Argumente über kontextfreie Sprachen.

Chomsky-Normalform (I)

Definition (Chomsky-Normalform)

Eine kontextfreie Grammatik $G = (V, \Sigma, P, S)$ ist in **Chomsky-Normalform**, wenn alle Regeln von G von einer der folgenden Formen sind:

$$A \to BC \quad \text{oder} \quad A \to a \quad \text{oder} \quad S \to \varepsilon \quad (A, B, C \in V, a \in \Sigma).$$

und falls $S \to \varepsilon$ vorhanden ist, kommt S auf keiner rechten Seite einer Regel vor.

Beispiel

Grammatik $G = (\{ S, X, Y, Z \}, \{ a, b, c \}, P, S)$ mit

$$\begin{array}{rrcl} P: & S & \to & SX \mid YZ \\ & X & \to & XY \mid b \mid c \\ & Y & \to & XZ \mid a \\ & Z & \to & a \mid c \end{array}$$

Chomsky-Normalform (II)

Satz 5.2

Zu jeder kontextfreien Grammatik G gibt es eine Grammatik G' in Chomsky-Normalform mit $L(G') = L(G)$.

- Beweis liefert Algorithmus zur Berechnung einer Chomsky-NF
- 3 wesentliche Schritte:
  1. Regeln der Form $A \to \varepsilon$ mit $A \neq S$ entfernen
  2. „Kettenregeln“ der Form $A \to B$ entfernen
  3. Regeln in Chomsky-NF-Regeln umformen

ε-freie kontextfreie Grammatiken

- Regeln der Form $A \to \varepsilon$ nicht nötig
- Außer $\varepsilon \in L(G)$; dann genügt: $S \to \varepsilon$.
- Noch etwas strikter (für beliebige Grammatiken):

Definition

Eine Grammatik $G = (V, \Sigma, P, S)$ heißt ε-**frei**, wenn gilt:

- G enthält keine Regel der Form $u \to \varepsilon$ mit $u \neq S$, und
- falls G die Regel $S \to \varepsilon$ enthält, kommt S auf keiner rechten Seite einer Regel vor; das heißt:

für alle $u \to v$ in P gilt: S kommt nicht in v vor.

Konstruktion ε-freier kontextfreier Grammatiken (I)

Gegeben: kontextfreie Grammatik $G = (V, \Sigma, P, S)$

Aufgabe: konstruiere ε-freie kontextfreie Grammatik

$$G_\varepsilon = (V_\varepsilon, \Sigma, P_\varepsilon, S_\varepsilon)$$

Algorithmus: (berechnet $U_\varepsilon = \{A \in V \mid A \Rightarrow^* \varepsilon\}$)

1) Bestimme die Menge $U_\varepsilon \subseteq V$ durch:
   a) $U_\varepsilon := \{ A \in V \mid A \to \varepsilon \text{ in } P \}$.
   b) Solange sich U_ε vergrößert:

$$U_\varepsilon := U_\varepsilon \cup \{A \in V \mid \text{es gibt } A \to A_1 \cdots A_k \text{ mit } A_1, \ldots, A_k \in U_\varepsilon \text{ in } P\}$$

2) Falls $S \in U_\varepsilon$:

$$V_\varepsilon := V \cup \{S'\}, \quad S_\varepsilon := S' \quad \text{(neues Startsymbol)}$$

Sonst $\quad V_\varepsilon := V, \quad S_\varepsilon := S$

Konstruktion ε-freier kontextfreier Grammatiken (II)

Algorithmus (Fortsetzung):

3) Bestimme die Regelmenge P_ε aus P:
   a) Entferne aus P alle Regeln der Form $B \to \varepsilon$.
   b) Für jede Regel $A \to xBy$ mit $B \in U_\varepsilon$ und $xy \neq \varepsilon$: füge neue Regel $A \to xy$ hinzu
   c) Wiederhole b) so lange, bis keine neuen Regeln mehr hinzukommen.
   d) Falls $S \in U_\varepsilon$: Regeln $S' \to \varepsilon \mid S$ hinzufügen.

Konstruktion ε-freier kontextfreier Grammatiken (III)

Beispiel

kontextfreie Grammatik $G = (\{S, A, B, C\}, \{a\}, P, S)$ mit

$$\begin{array}{lcllcl} P: S & \to & ABC \mid B, & B & \to & \varepsilon, \\ A & \to & a, & C & \to & AS \end{array}$$

- Schritt 1: $U_\varepsilon = \{B, S\}$
- Schritt 2: $V_\varepsilon = \{S', S, A, B, C\}$ und $S_\varepsilon = S'$
- Schritt 3: P_ε berechnen
  - entferne $B \to \varepsilon$ aus P
  - füge neue Regeln hinzu:

$$S \to AC, \quad C \to A \quad \text{sowie} \quad S' \to \varepsilon \mid S$$

Konstruktion ε-freier kontextfreier Grammatiken (IV)

Beispiel (Fortsetzung)

Ergebnis: kontextfreie Grammatik

$$G_\varepsilon = (\{S', S, A, B, C\}, \{a\}, P_\varepsilon, S') \quad \text{mit}$$

$$\begin{array}{llllllll} P_\varepsilon: & S' & \to & \varepsilon \mid S, & & A & \to & a, \\ & S & \to & ABC \mid B \mid AC, & & C & \to & AS \mid A. \end{array}$$

Zusammenhang kontextfreier und ε-freier kontextfreier Grammatiken

Allgemein gilt:

Lemma 5.3

Die Grammatik G_ε ist kontextfrei und ε-frei, und es gilt $L(G_\varepsilon) = L(G)$.

Daraus folgt direkt:

Satz 5.4

Zu jeder kontextfreien Grammatik G gibt es eine ε-freie kontextfreie Grammatik G' mit $L(G') = L(G)$.

ε-freie kontextfreie Grammatiken ohne Kettenregeln (I)

Als nächstes:

Kettenregeln der Art $A \to B$ mit $A, B \in V$ eliminieren.

Gegeben: ε-freie, kontextfreie Grammatik $G = (V, \Sigma, P, S)$

Aufgabe: konstruiere ε-freie, kontextfreie Grammatik

$$G_{oK} = (V, \Sigma, P_{oK}, S) \text{ ohne Kettenregeln}$$

Algorithmus:

1) Entferne Regel-„Zyklen“ $(A_1, \ldots, A_k \in V)$
   $A_1 \to A_2, \quad A_2 \to A_3, \quad \ldots, \quad A_k \to A_1$:
   a) ersetze alle $A_2, \ldots, A_k$ durch A_1 in allen Regeln von P und entferne alle Regeln der Form $C \to C$.
   b) Wiederhole Schritt *a*) so lange, bis keine derartigen Regel-Zyklen mehr vorhanden sind.
2) Bestimme Nummerierung $B_1, \ldots, B_n$ der Elemente von V mit:

$$\text{falls } B_i \to B_j \in P, \text{ dann } i < j.$$

ε-freie kontextfreie Grammatiken ohne Kettenregeln (II)

Algorithmus (Fortsetzung):

3) Für $k = n-1, \ldots, 2, 1$ und alle $j > k$:

   entferne Regeln der Form $B_k \to B_j$,

   falls $B_j \to x$ vorhanden, füge $B_k \to x$ hinzu

   („rückwärts einsetzen“ rechter Seiten von B_j)

Es gilt:

Lemma 5.5

Die Grammatik G_{oK} ist ε-frei, kontextfrei, enthält keine Kettenregeln, und es gilt $L(G_{oK}) = L(G)$.

ε-freie kontextfreie Grammatiken ohne Kettenregeln (III)

Beispiel

Von vorher: $G_\varepsilon = (\{S', S, A, B, C\}, \{a\}, P_\varepsilon, S')$ mit

$$\begin{array}{llllll} P_\varepsilon : S' & \to & \varepsilon \mid S, & A & \to & a, \\ S & \to & ABC \mid B \mid AC, & C & \to & AS \mid A. \end{array}$$

- Schritt 1: keine Zyklen vorhanden $\rightsquigarrow$ keine Änderung
- Schritt 2: betrachte Regeln $S' \to S \to B$ und $C \to A$:
  Numerierung: $S' \rightsquigarrow 1$, $S \rightsquigarrow 2$, $B \rightsquigarrow 3$, $C \rightsquigarrow 4$, $A \rightsquigarrow 5$
- Schritt 3: entferne: $S' \to S$, $S \to B$ und $C \to A$
  füge hinzu: $S' \to AC \mid ABC$ und $C \to a$,
  (da G_ε die Regeln $S \to AC \mid ABC$ und $A \to a$ enthält)

Ergebnisregeln: $\begin{array}{llllll} S' & \to & \varepsilon \mid AC \mid ABC, & A & \to & a, \\ S & \to & AC \mid ABC, & C & \to & AS \mid a. \end{array}$

Chomsky-Normalform-Algorithmus (I)

Gegeben: beliebige kontextfreie Grammatik $G = (V, \Sigma, P, S)$.

Aufgabe: Konstruiere Grammatik $G_{CNF} = (V_{CNF}, \Sigma, P_{CNF}, S)$ in Chomsky-Normalform.

Algorithmus:

1.) Konstruiere aus G gemäß den obigen Verfahren eine ε-freie kontextfreie Grammatik G' ohne Kettenregeln.

2.) Für jedes $a \in \Sigma$:
füge neue Regel $C_a \to a$ hinzu (C_a neue Variable)
ersetze a durch C_a in Regeln $A \to x$ mit $|x| \geq 2$

3.) Für jede Regel $A \to B_1 B_2 \dots B_n$ (mit $B_1, \dots, B_n \in V'$, $n > 2$)
ersetze die Regel durch

$$A \to B_1 D_1, \quad D_1 \to B_2 D_2, \quad \dots, \quad D_{n-2} \to B_{n-1} B_n.$$

($D_1, \dots, D_{n-2}$ neue Variablen)

Chomsky-Normalform (III)

Beispiel

- Sei G die Grammatik mit den Regeln

$$S \to bA \mid aB, \qquad B \to aBB \mid bS \mid b.$$
$$A \to bAA \mid aS \mid a,$$

- G ist bereits ε-frei und hat keine Kettenregeln.
- Schritt 2 liefert:

$$S \to C_bA \mid C_aB, \qquad C_a \to a,$$
$$A \to C_bAA \mid C_aS \mid a, \qquad C_b \to b.$$
$$B \to C_aBB \mid C_bS \mid b,$$

- Schritt 3 liefert:

$$S \to C_bA \mid C_aB,$$
$$A \to C_bD \mid C_aS \mid a, \qquad D \to AA,$$
$$B \to C_aE \mid C_bS \mid b, \qquad E \to BB.$$
$$C_a \to a, \quad C_b \to b,$$

Chomsky-Normalform (IV)

Es gilt:

Lemma 5.5

Die Grammatik G_{CNF} ist in Chomsky-Normalform, und es gilt $L(G_{CNF}) = L(G)$.

Daraus folgt direkt:

Satz 5.6

Zu jeder kontextfreien Grammatik G gibt es eine Grammatik G' in Chomsky-Normalform mit $L(G') = L(G)$.

Greibach-Normalform (I)

- Die Chomsky-Normalform ist für theoretische und algorithmische Betrachtungen von kontextfreien Grammatiken hilfreich.
- Es gibt noch weitere Normalformen kontextfreier Grammatiken.
- Wichtig für die praktische Anwendung bei der Syntaxanalyse von Programmen:

Definition

Eine kontextfreie Grammatik $G = (V, \Sigma, P, S)$ ist in **Greibach-Normalform**, wenn sie ε-frei ist und wenn alle Regeln von G (mit der eventuellen Ausnahme von $S \to \varepsilon$) von der folgenden Form sind:

$$A \to aB_1 \cdots B_k \qquad (A, B_i \in V, a \in \Sigma, k \geq 0).$$

Greibach-Normalform (II)

Es gilt auch hier:

Satz 5.7

Zu jeder kontextfreien Grammatik G gibt es eine Grammatik G' in Greibach-Normalform mit $L(G') = L(G)$.

Entscheidbarkeit kontextfreier Sprachen

Entscheidbarkeit kontextfreier Sprachen (I)

Jetzt: jede kontextfreie Sprache ist entscheidbar.

Es gilt noch stärker:

Satz 5.8

Das allgemeine Wortproblem für kontextfreie Sprachen ist entscheidbar.

Allgemeines Wortproblem für kontextfrei Sprachen

Eingabe: (Code einer) kontextfreien Grammatik $G = (V, \Sigma, S, P)$, Wort $w \in \Sigma^*$

Aufgabe: Entscheiden, ob $w \in L(G)$ gilt.

Entscheidbarkeit kontextfreier Sprachen (II)

CYK-Algorithmus
Algorithmus von John **C**ocke, Daniel **Y**ounger und Tadao **K**asami

Effizienter Algorithmus für das allgemeine Wortproblem:

Eingabe:

- kontextfreie Grammatik $G = (V, \Sigma, P, S)$ in Chomsky-NF
- Wort $w \in \Sigma^*$

Aufgabe: entscheiden, ob $w \in L(G)$ gilt
(und ggf. Ableitung berechnen)

Der Cocke-Younger-Kasami-/CYK-Algorithmus (I)

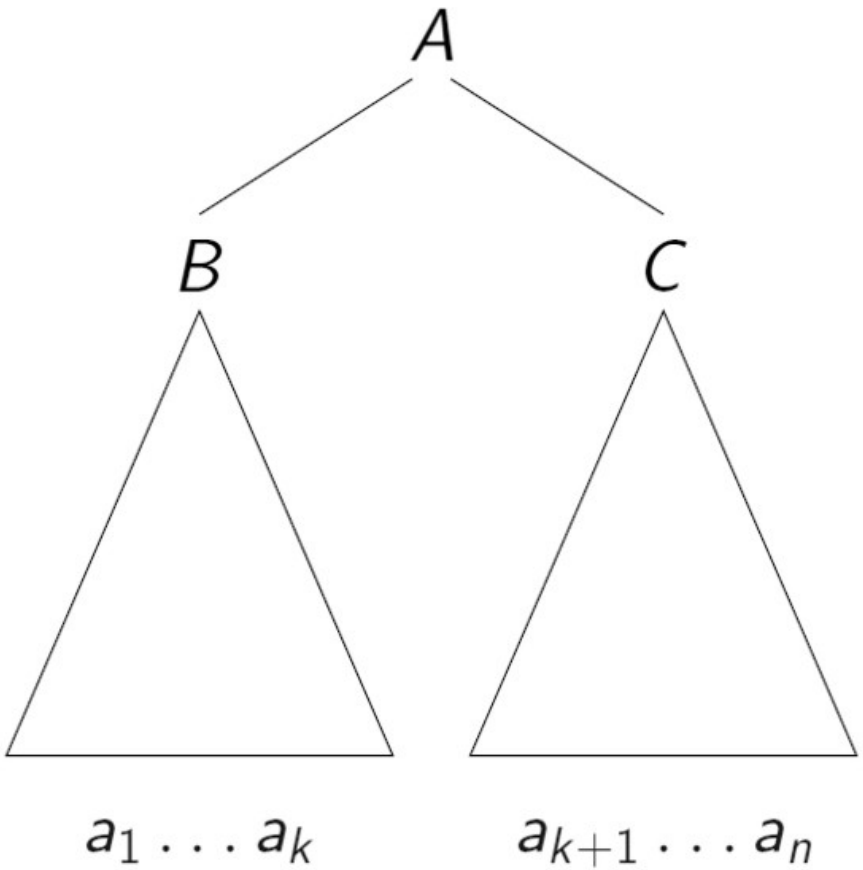
Idee:

- Versuche für $w \in \Sigma^*$ rekursiv einen Ableitungsbaum „rückwärts“ auszurechnen
- **Anfang:** $w = a \in \Sigma$;
  w kann nur abgeleitet werden durch Regel $A \to a$
- **Rekursionsschritt:** $w = a_1 \cdots a_n$, $n \geq 2$.
  w kann nur wie folgt abgeleitet werden:
  1. eine Regel $A \to BC$ anwenden
  2. aus B das Teilwort $a_1 a_2 \cdots a_k$ ableiten
  3. aus C das Teilwort $a_{k+1} \cdots a_n$ ableiten

  (Hierbei gilt $1 \leq k < n$.)

Der Cocke-Younger-Kasami-/CYK-Algorithmus (II)

Rekursionsschritt



Der Cocke-Younger-Kasami-/CYK-Algorithmus (III)

Im **Rekursionsschritt**:

- alle möglichen $k = 1, 2, \ldots, n-1$ probieren
- alle möglichen Regeln $A \to BC$ probieren

Rekursive Prozedur:

Eingabe: $x = a_1 \cdots a_n$

Rückgabe: Menge V der Variablen A mit $A \Rightarrow^* x$

1. Falls $x = a \in \Sigma$, gib zurück: $V = \{ A \mid A \to a \text{ in } P \}$
2. Sonst: für alle $k = 1, 2, \ldots n-1$:
   (für jede Teilung $x = a_1 \cdots a_k \mid a_{k+1} \cdots a_n$)
   1. bestimme Menge V_1 aller Variablen B mit $B \Rightarrow^* a_1 \ldots a_k$
      (rekursiver Aufruf)
   2. bestimme Menge V_2 aller Variablen C mit $C \Rightarrow^* a_{k+1} \ldots a_n$
      (rekursiver Aufruf)
   3. gib zurück: $V := \{ A \mid A \to BC \text{ mit } B \in V_1 \text{ und } C \in V_2 \}$
      (alle Kombinationen BC betrachten und linke Seiten finden)

Dynamische Programmierung

CYK-Algorithmus benutzt **dynamische Programmierung**
(statt obige Rekursion zu implementieren)

- Mengen V_1 und V_2 werden in einer Tabelle gespeichert
  (zur Wiederverwendung in folgenden „Rekursionsschritten“)
- Mengen in der Tabelle werden systematisch iterativ berechnet:
  1. berechne alle Variablen, aus denen sich Teilwörter von x der Länge 1 ableiten lassen,
  2. berechne dann alle Variablen, aus denen sich Teilwörter von x der Länge 2 ableiten lassen,
  3. ...
  4. zuletzt berechne alle Variablen, aus denen sich x ableiten lässt.
- **Am Schluss:** falls das Startsymbol S unter diesen Variablen, so liegt x in $L(G)$.

Der Cocke-Younger-Kasami-/CYK-Algorithmus (V)

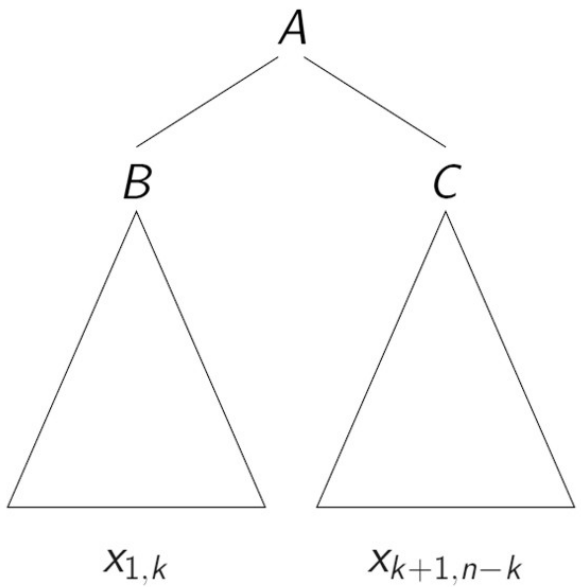
Notation

Für $x = a_1 \dots a_n$ ist

$$x_{i,j} = a_i a_{i+1} \cdots a_{i+j-1}$$

Teilwort von x, das an Stelle i beginnt und Länge j hat.

Obiges Bild:



Der Cocke-Younger-Kasami-/CYK-Algorithmus (VI)

Notation

Für $x = a_1 \dots a_n$ ist

$$T_{i,j} = \{ A \in V \mid A \Rightarrow^* x_{i,j} \}$$

Menge der Variablen, aus denen $x_{i,j}$ ableitbar ist.

Berechnung von $T_{i,j}$ aus Mengen $T_{i',j'}$ mit $j' < j$:

$$T_{i,j} = \{A \mid (A \to BC) \in P \text{ und es gibt } k = 1, \dots j-1 \text{ mit } B \in T_{i,k} \quad \text{und} \quad C \in T_{i+k,j-k} \}$$

Der Cocke-Younger-Kasami-/CYK-Algorithmus (VII)

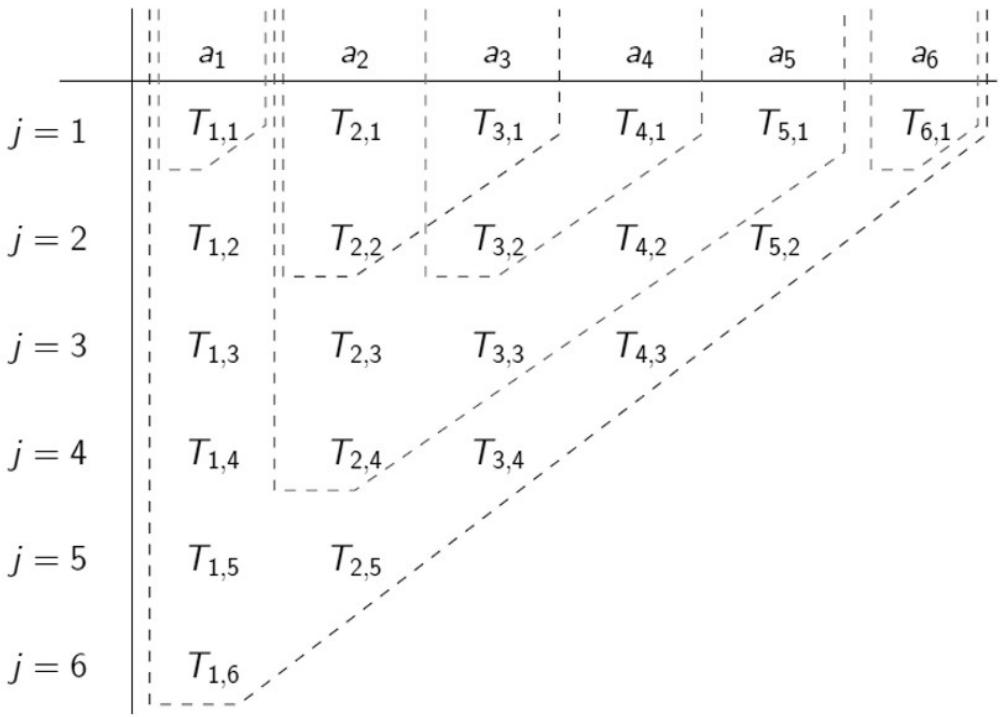
Implementierung des CYK-Algorithmus:

Variablenmengen $T_{i,j}$ werden in folgender Tabelle berechnet/gespeichert:

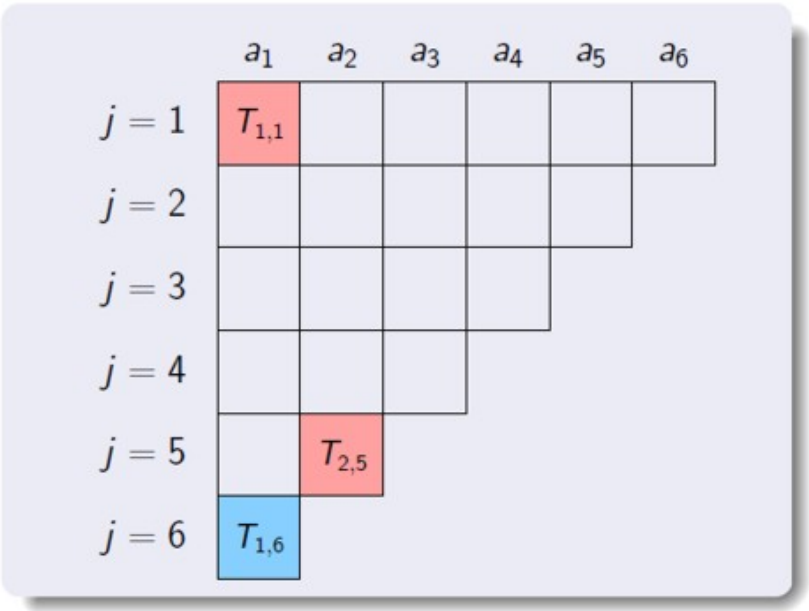
	a_1	a_2	$\dots$		a_{n-1}	a_n
$j=1$	$T_{1,1}$	$T_{2,1}$	$\dots$	$\dots$	$T_{n-1,1}$	$T_{n,1}$
$j=2$	$T_{1,2}$	$T_{2,2}$	$\dots$	$\dots$	$T_{n-1,2}$	
	$\dots$	$\dots$	$\dots$	$\dots$		
$\dots$	$\dots$	$\dots$	$\dots$			
$j=n-1$	$T_{1,n-1}$	$T_{2,n-1}$				
$j=n$	$T_{1,n}$					

Der Cocke-Younger-Kasami-/CYK-Algorithmus (VIII)

Ableitung von Teilwörtern durch Variablen:



Der Cocke-Younger-Kasami-/CYK-Algorithmus (IX)

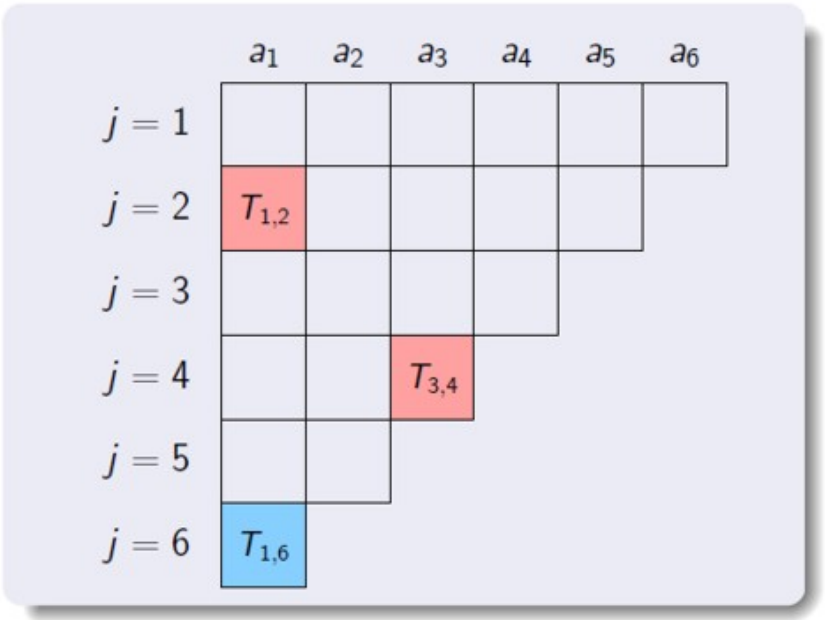


$x = a_1 \mid a_2 a_3 a_4 a_5 a_6$

$(A \to BC) \in P$,
$B \in T_{1,1}$, $C \in T_{2,5} \Rightarrow A \in T_{1,6}$

Ziel: Bestimmung von $T_{1,6}$.
Durchlaufe alle möglichen Werte von k.
Hier: $k = 1$.

Der Cocke-Younger-Kasami-/CYK-Algorithmus (X)

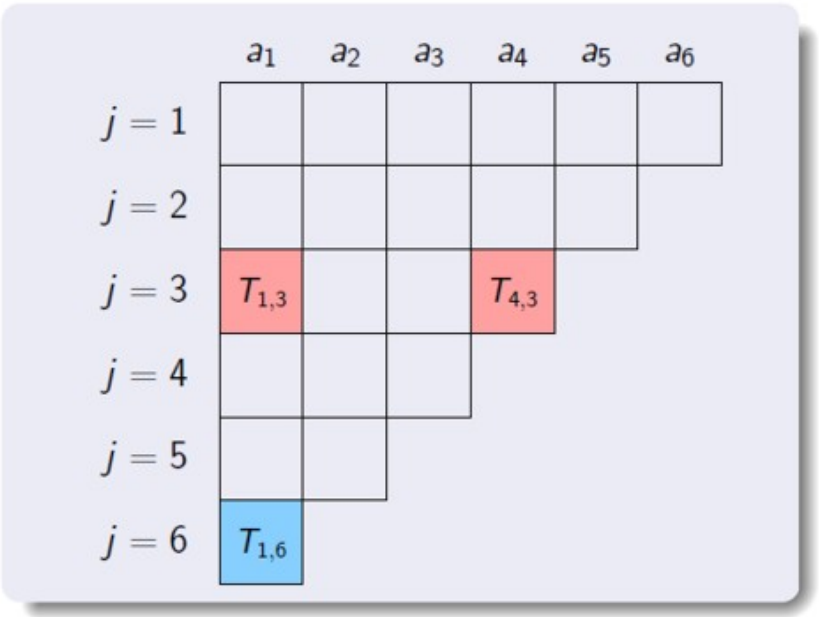


$x = a_1 a_2 \mid a_3 a_4 a_5 a_6$

$(A \to BC) \in P$,
$B \in T_{1,2}$, $C \in T_{3,4} \Rightarrow A \in T_{1,6}$

Ziel: Bestimmung von $T_{1,6}$.
Durchlaufe alle möglichen Werte von k.
Hier: $k = 2$.

Der Cocke-Younger-Kasami-/CYK-Algorithmus (XI)

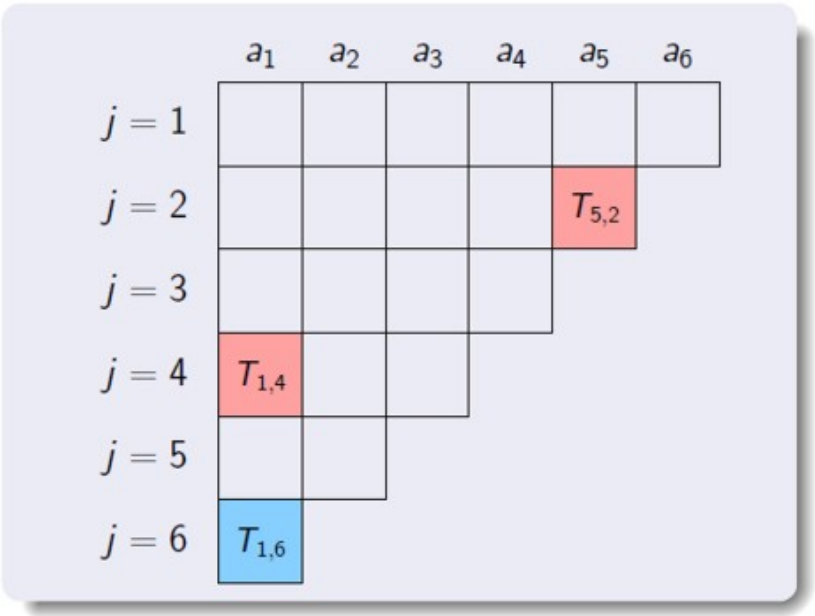


$x = a_1 a_2 a_3 \mid a_4 a_5 a_6$

$(A \to BC) \in P$,
$B \in T_{1,3}$, $C \in T_{4,3} \Rightarrow A \in T_{1,6}$

Ziel: Bestimmung von $T_{1,6}$.
Durchlaufe alle möglichen Werte von k.
Hier: $k = 3$.

Der Cocke-Younger-Kasami-/CYK-Algorithmus (XII)

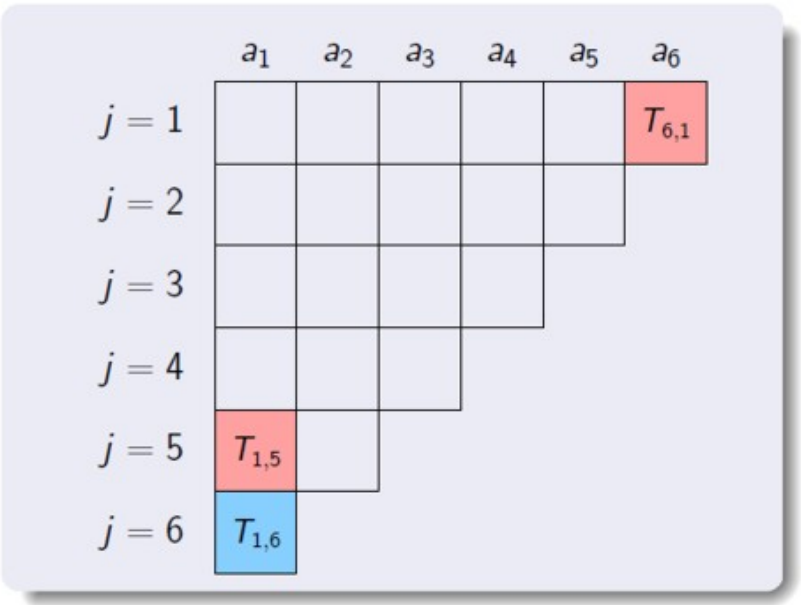


$x = a_1 a_2 a_3 a_4 \mid a_5 a_6$

$(A \to BC) \in P,$
$B \in T_{1,4},\ C \in T_{5,2} \Rightarrow A \in T_{1,6}$

Ziel: Bestimmung von $T_{1,6}$.
Durchlaufe alle möglichen Werte von k.
Hier: $k = 4$.

Der Cocke-Younger-Kasami-/CYK-Algorithmus (XII)



$x = a_1 a_2 a_3 a_4 a_5 \mid a_6$

$(A \to BC) \in P,$
$B \in T_{1,5},\ C \in T_{6,1} \Rightarrow A \in T_{1,6}$

Ziel: Bestimmung von $T_{1,6}$.
Durchlaufe alle möglichen Werte von k.
Hier: $k = 5$.

Der Cocke-Younger-Kasami-/CYK-Algorithmus (XIII)

Beispiel

Gegeben: kontextfreie Grammatik $G = (\{S, A, B, C\}, \{a, b, \}, P, S)$

$$\begin{aligned} P: S &\to AB \mid BC \\ A &\to BA \mid a \\ B &\to CC \mid b \\ C &\to AB \mid a \end{aligned}$$

Frage: Gilt $baaba \in L(G)$?
CYK-Tabelle berechnen:

	b	a	a	b	a
$j = 1$	B	A, C	A, C	B	A, C

Erste Zeile ($j = 1$): betrachte Regeln $X \to s \quad X \in V, s \in \Sigma$.

Der Cocke-Younger-Kasami-/CYK-Algorithmus (XIV)

Beispiel (Fortsetzung)

	b	a	a	b	a
$j=1$	B	A, C	A, C	B	A, C
$j=2$	S, A	B	S, C	S, A	

Ab zweite Zeile ($j \geq 2$):

Regeln $X \to YZ \quad X, Y, Z \in V$ betrachten.

Berechnung von $T_{1,2}$ (analog $T_{2,2}$, $T_{3,2}$, $T_{4,2}$):

- $k = 1$ (einziger möglicher Wert)
- Variablen aus $T_{1,1}$ und $T_{2,1}$ kombinieren (Reihenfolge beachten!):
  $$T_{1,1}\,T_{2,1} = \{BA, BC\}$$
- mögliche linke Seiten: S, A

  (Regeln: $S \to BC$ und $A \to BA$)

Der Cocke-Younger-Kasami-/CYK-Algorithmus (XV)

Beispiel (Fortsetzung)

	b	a	a	b	a
$j=1$	B	A, C	A, C	B	A, C
$j=2$	S, A	B	S, C	S, A	
$j=3$	$-$	B	B		

Berechnung von $T_{2,3}$ (analog: $T_{1,3}$, $T_{3,3}$):

- Teilwort aab an Positionen 2–4 $\rightsquigarrow$ $k = 1, 2$ möglich:

$$(a)(ab) \quad \text{oder} \quad (aa)(b)$$

- also forme: $T_{2,1}\,T_{3,2} \cup T_{2,2}\,T_{4,1} = \{\, AS, AC, CS, CC, BB \,\}$
- mögliche linke Seiten: B (wegen Regel $B \to CC$)

Der Cocke-Younger-Kasami-/CYK-Algorithmus (XVI)

Beispiel (Fortsetzung)

Ergebnis:

	b	a	a	b	a
$j = 1$	B	A, C	A, C	B	A, C
$j = 2$	S, A	B	S, C	S, A	
$j = 3$	$-$	B	B		
$j = 4$	$-$	S, A, C			
$j = 5$	S, A, C				

Da $S \in T_{1,5}$ liegt, gilt $baaba \in L(G)$.

Der Cocke-Younger-Kasami-/CYK-Algorithmus (XVII)

Beispiel (Fortsetzung)

Ergebnis:

	b	a	a	b	a
$j=1$	B	A, C	A, C	B	A, C
$j=2$	S, A	B	S, C	S, A	
$j=3$	–	B	B		
$j=4$	–	S, A, C			
$j=5$	S, A, C				

Da $S \in T_{1,5}$ liegt, gilt $baaba \in L(G)$.

Ableitungsbaum für $baaba$:

durch „Zurückverfolgung“ des Startsymbols S aus $T_{1,5}$

Der Cocke-Younger-Kasami-/CYK-Algorithmus (XVIII)

input $G = (V, \Sigma, P, S)$, $w \in \Sigma^*$
$n := |w|$
for $i \in \{1, \ldots, n\}$ do
  $T_{i,1} := \{A \mid A \to x_{j,1} \in P\}$
for $j \in \{2, \ldots, n\}$ do
  for $i \in \{1, \ldots, n - j + 1\}$ do
    $T_{i,j} := \varnothing$
    for $k \in \{1, \ldots, j - 1\}$ do
      $T_{i,j} := T_{i,j} \cup$
        $\{A \mid A \to BC \in P$ for some $B \in T_{i,k}, C \in T_{i+k,j-k}\}$
    end
  end
end
if $S \in T_{1,n}$ then return true else return false

Pumping-Lemma für kontextfreie Sprachen

Das Pumping-Lemma für kontextfreie Sprachen (I)

Ideen

- Pumping-Lemma für **reguläre Sprachen**:

  „Jedes ausreichend lange Wort durchläuft einen Zustand eines DFA zweimal.“

- Pumping-Lemma für **kontextfreie Sprachen**:

  „Auf einem Pfad des Ableitungsbaums eines ausreichend langen Wortes, kommt eine Variable mindestens zweimal vor.“

Das Pumping-Lemma für kontextfreie Sprachen (II)

Frage: Was bedeutet hier „ausreichend langes Wort“?

Beobachtung:

Für Chomsky-NF Grammatiken gilt aufgrund der Regeln $A \to BC$:

Ableitungsbäume sind Binärbäume
(außer unterste Schicht vor den Blättern)

Lemma 5.9

Sei B ein Binärbaum mit mindestens 2^k Blättern.

Dann hat B Pfad, der aus mindestens k Kanten besteht.
(und $k+1$ Knoten)

Das Pumping-Lemma für kontextfreie Sprachen (III)

Daraus folgt für eine Grammatik $G = (V, \Sigma, P, S)$ in Chomsky-NF:

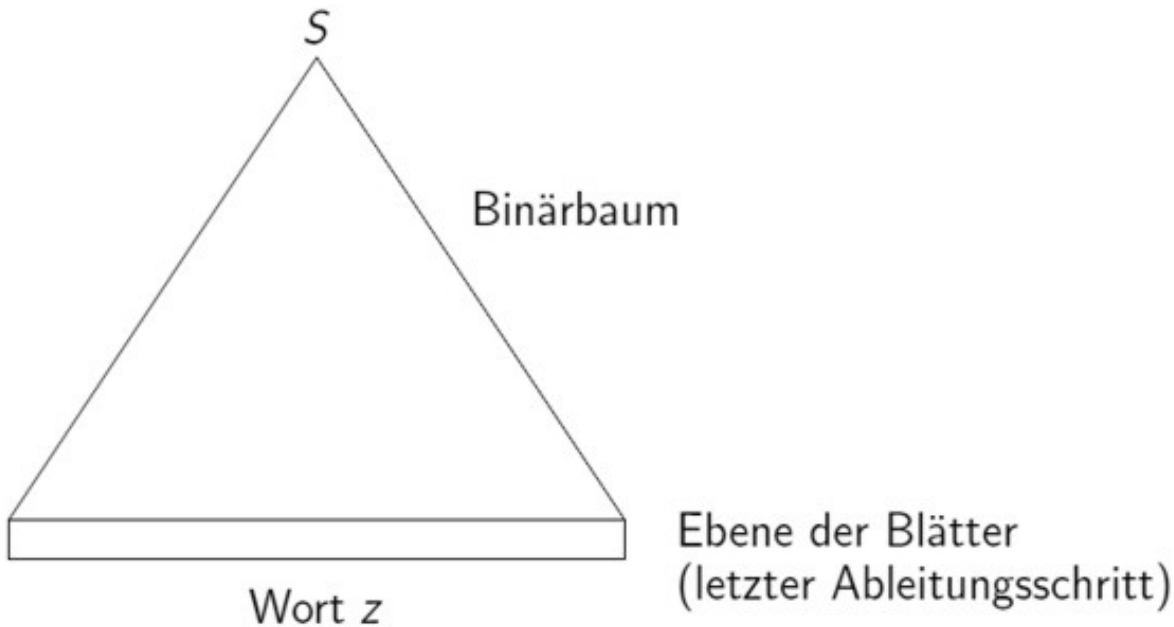
- Sei k die Anzahl von Variablen in G ($k = |V|$).
  Für $z \in L$ mit $|z| \geq 2^k$ hat jeder Ableitungsbaum $\geq 2^k$ Blätter.
- Das gilt auch nach Abschneiden aller Blätter (Regeln $A \to a$)
  $\Longrightarrow$ der beschnittene Baum hat einen Pfad mit $\geq k + 1$ Knoten
- Betrachte einen maximalen Pfad

  $$S = A_1 \to A_2 \to \cdots \to A_\ell \to a \qquad A_1, \ldots, A_\ell \in V, a \in \Sigma.$$

  Da $\ell \geq k + 1$ gilt, existieren i, j mit $1 \leq i < j \leq \ell$ und $A_i = A_j$.
- Also muss auf diesem Pfad eine Variable A zweimal vorkommen.

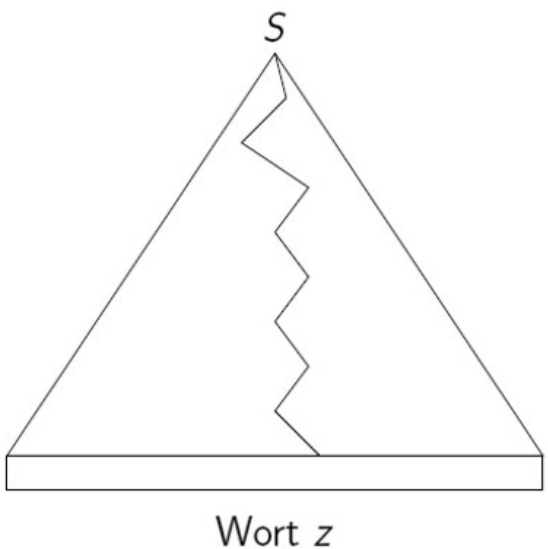
Das Pumping-Lemma für kontextfreie Sprachen (IV)

Ableitungsbaum für ein Wort z mit $|z| \geq n = 2^k$
n ist hier die „Konstante des Pumping-Lemmas“

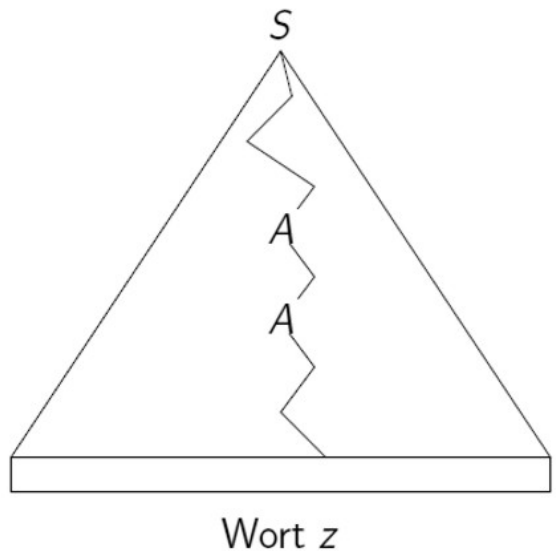


Das Pumping-Lemma für kontextfreie Sprachen (V)

Es gibt einen Pfad mit mindestens $k + 1$ inneren Knoten.



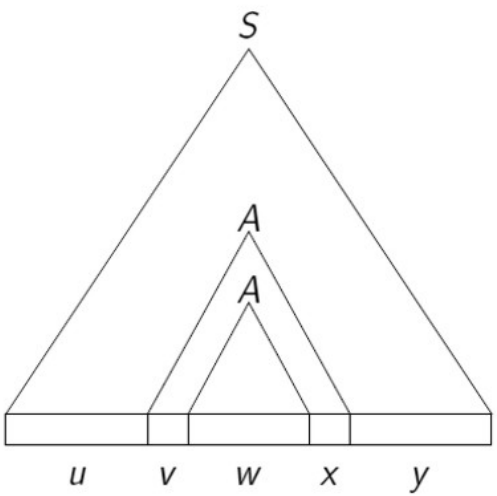
Auf diesem Pfad gibt es eine Variable A, die zweimal auftaucht.



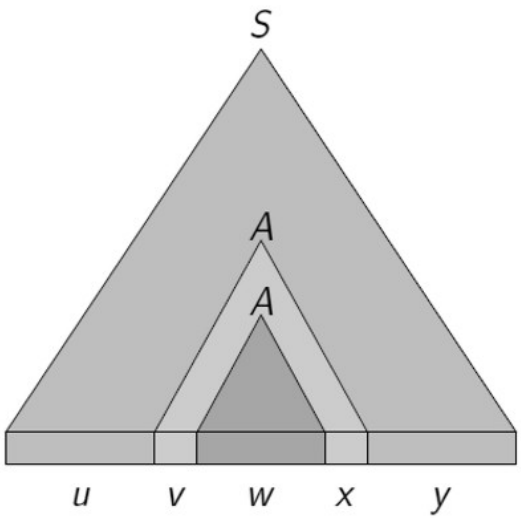
Das Pumping-Lemma für kontextfreie Sprachen (VI)

Das Wort z wird nun in fünf Teilwörter u, v, w, x, y zerlegt:

- w wird aus dem unteren A abgeleitet: $A \Rightarrow^* w$
- vwx wird aus dem oberen A abgeleitet: $A \Rightarrow^* vwx$



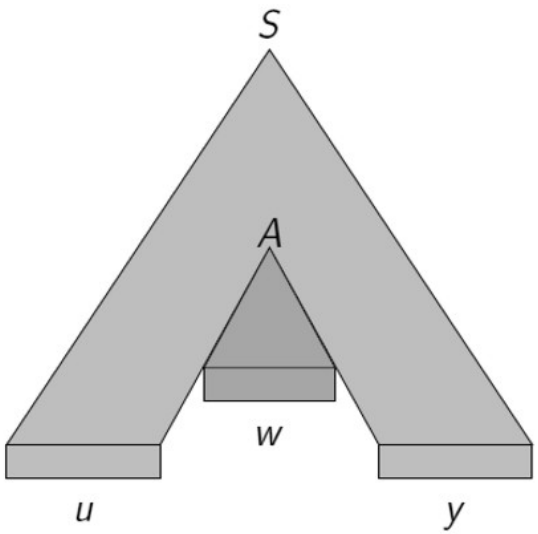
Drei Teil-Syntaxbäume, die man neu „zusammenstecken“ kann.



Das Pumping-Lemma für kontextfreie Sprachen (VII)

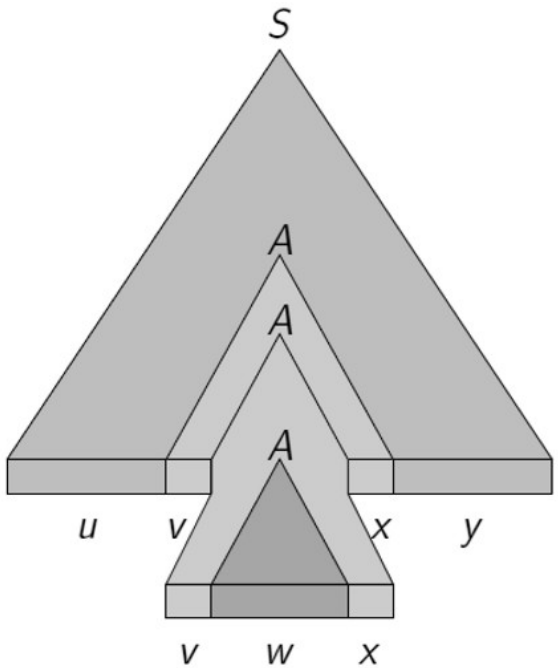
Weglassen des mittleren Teilbaums
$\rightsquigarrow$ Ableitungsbaum für uwy.

Damit gilt: $uwy \in L$.



Verdoppeln des mittleren Teilbaums
$\rightsquigarrow$ Ableitungsbaum für uv^2wx^2y.

Damit gilt: $uv^2wx^2y \in L$.
Allgemein: $uv^kwx^ky \in L$.



Das Pumping-Lemma für kontextfreie Sprachen (VIII)

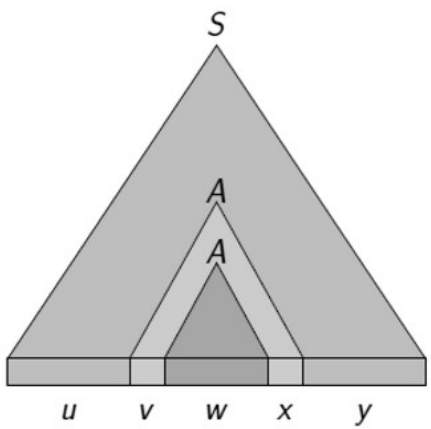
Außerdem haben v, w, x folgende Eigenschaften:

1) $|vwx| \leq n = 2^k$:

   Wähle Doppelvorkommen mit der größten Tiefe
   (durchsuche Pfad von unten nach oben)

   $\Longrightarrow$ Abstand oberes A $\rightsquigarrow$ Blatt $\leq k$ (= Anzahl Kanten)

   $\Longrightarrow$ der Baum für vwx hat höchstens 2^k Blätter



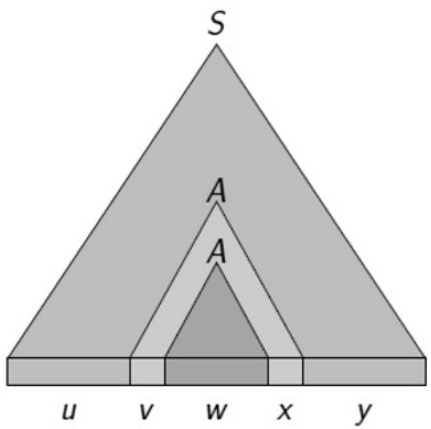
Das Pumping-Lemma für kontextfreie Sprachen (IX)

2) $|vx| \geq 1$:

Betrachte die Kinder B, C des oberen A
(Regel $A \to BC$ angewendet)

unteres A wird entweder aus B oder C abgeleitet

wegen Chomsky-NF muss die andere Variable mindestens ein Terminalsymbol erzeugen
(das entweder in v oder x vorkommt)



Das Pumping-Lemma für kontextfreie Sprachen (X)

Satz 5.10 (Pumping-Lemma für kontextfreie Sprachen)

Sei L eine kontextfreie Sprache über einem Alphabet Σ.

Dann gibt es eine natürliche Zahl $n \in \mathbb{N}$, so dass gilt:

Zu jedem $z \in L$ mit $|z| \geq n$ gibt es $u, v, w, x, y \in \Sigma^*$ mit

$$z = uvwxy \qquad \text{und}$$

1. $|vx| \geq 1$,
2. $|vwx| \leq n$,
3. Für jedes $k \in \mathbb{N}$ gilt: $uv^k wx^k y \in L$.

n heißt auch **Pumping-Zahl** für L.

Das Pumping-Lemma für kontextfreie Sprachen (XI)

Anwendung des Pumping-Lemmas:

Beweisen, dass eine gegebene formale Sprache L nicht kontextfrei ist.

Kochrezept

Gegeben: formale Sprache $L \subseteq \Sigma^*$

1. Nimm eine beliebige Zahl n an.
2. Wähle ein Wort $z \in L$ mit $|z| \geq n$.
3. Betrachte alle möglichen Zerlegungen $z = uvwxy$ mit

$$|vx| \geq 1 \quad \text{und} \quad |vwx| \leq n.$$

4. Zeige, dass es für alle Zerlegungen eine Zahl i gibt mit

$$uv^i wx^i y \notin L.$$

Dann ist L **nicht** kontextfrei.

Das Pumping-Lemma für kontextfreie Sprachen (XII)

Beispiel

Gegeben: $L = \{a^i b^i c^i \in \{a, b, c\}^* \mid i \in \mathbb{N}\}$

- Annahme: L ist kontextfrei. Sei n Pumping-Zahl.
- Wähle $z = a^n b^n c^n$.
- Es gilt $|z| \geq n$. Also existiert Zerlegung
  $$z = uvwxy$$
  mit den in Satz 5.10 genannten Eigenschaften 1-3.
- Wegen $|vwx| \leq n$:
  vwx kann nicht zugleich Zeichen a, b und c enthalten.
- Wegen $|vx| \geq 1$: vx (somit auch uwy) kann nicht eine gleiche Anzahl an Zeichen a, b und c enthalten.
- Dann gilt: $uv^0wx^0y = uwy \notin L$. Widerspruch!

Zusammenfassung