Technische
Universität
Dresden

Formale Systeme

13. Vorlesung: Das Pumping Lemma kontextfreier Sprachen

Markus Krötzsch

Professur für Wissensbasierte Systeme

TU Dresden, 27. November 2025

CYK: Grundidee

Der CYK-Algorithmus arbeitet mit einer kontextfreien Grammatik G in CNF.

Wie kann man prüfen, ob ein Wort $w = \mathtt{a}_1 \cdots \mathtt{a}_n$ durch so eine Grammatik abgeleitet werden kann?

- Falls $|w| = 1$, dann ist $w \in \Sigma$ und es gilt:
  $w \in \mathbf{L}(G)$ genau dann wenn es eine Regel $\mathsf{S} \to w$ in G gibt
- Falls $|w| > 1$, dann ist:
  $w \in \mathbf{L}(G)$ genau dann wenn es eine Regel $\mathsf{S} \to \mathsf{AB}$ und eine Zahl i gibt, so dass gilt

$$\mathsf{A} \Rightarrow^* \mathtt{a}_1 \cdots \mathtt{a}_i \qquad \text{und} \qquad \mathsf{B} \Rightarrow^* \mathtt{a}_{i+1} \cdots \mathtt{a}_n$$

Idee: Fall 2 reduziert das Problem $\mathsf{S} \overset{?}{\Rightarrow}{}^* w$ in zwei einfachere Probleme $\mathsf{A} \overset{?}{\Rightarrow}{}^* \mathtt{a}_1 \cdots \mathtt{a}_i$ und $\mathsf{B} \overset{?}{\Rightarrow}{}^* \mathtt{a}_{i+1} \cdots \mathtt{a}_n$, die man allerdings für alle Regeln $\mathsf{S} \to \mathsf{AB}$ und Indizes i lösen muss

CYK: Praktische Umsetzung

Notation: Für $w = \mathtt{a}_1 \cdots \mathtt{a}_n$ schreiben wir $w_{i,j}$ für das Teilwort $\mathtt{a}_i \cdots \mathtt{a}_j$, also das Infix der Länge $j - i + 1$, welches an Position i beginnt.

Vorgehen: Wir berechnen für jedes Teilwort $w_{i,j}$ die Menge aller Variablen A, für die gilt A $\Rightarrow^* w_{i,j}$. Diese Menge nennen wir $V[i, j]$.

- Wir beginnen mit den kürzesten Teilwörtern ($i = j$)
- Für längere Wörter betrachten wir jede mögliche Zweiteilung $w_{i,\ell} = w_{i,k}w_{k+1,j}$ und suchen Regeln der Form A $\to$ BC so dass B $\in V[i, k]$ und C $\in V[k + 1, j]$

Ist am Ende das Startsymbol $S \in V[1, |w|]$, dann liegt w in der Sprache

Notation

Für die Teilwörter $w_{i,j}$ gilt $i \leq j$
$\leadsto$ Die Mengen $V[i,j]$ können als Dreiecksmatrix notiert werden

Beispiel: Wir betrachten das Wort $w = \mathtt{a} + \mathtt{b} * \mathtt{c}$ der Länge $|w| = 5$.

Darstellung der Mengen $V[i,j]$:

a	$V[1,1]$	$V[1,2]$	$V[1,3]$	$V[1,4]$	$V[1,5]$
+		$V[2,2]$	$V[2,3]$	$V[2,4]$	$V[2,5]$
b			$V[3,3]$	$V[3,4]$	$V[3,5]$
*				$V[4,4]$	$V[4,5]$
c					$V[5,5]$
	a	+	b	*	c

Beispiel

Grammatik:

S → SA | SM | a | b | c
A → PS   M → TS   P → +   T → $*$

Wort: w = a + b $*$ c

Berechnung der Mengen $V[i, j]$:

	a	+	b	$*$	c
a	S		S		S
+		P	A		A
b			S		S
$*$				T	M
c					S

$S \in V[1, 5] \rightsquigarrow w$ kann erzeugt werden

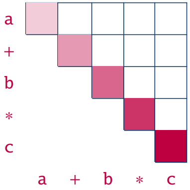
Der CYK-Algorithmus

Eingabe: Wort $w = \mathtt{a}_1 \cdots \mathtt{a}_n$,
CNF-Grammatik $G = \langle V, \Sigma, P, \mathsf{S} \rangle$
Ausgabe: „ja“ wenn $w \in \mathbf{L}(G)$; sonst „nein“

for $i = 1, \ldots, n$: // Teilwörter der Länge 1
  $V[i, i] := \{\mathsf{A} \in V \mid \mathsf{A} \to \mathtt{a}_i \in P\}$

for $d = 1, \ldots, n-1$: // Differenz Endind. – Startind.
  for $i = 1, \ldots, n-d$: // Startindex
    $j := i + d$
    $V[i, j] := \emptyset$
    for $k = i, \ldots, j-1$: // Trennindex
      $V[i, j] := V[i, j] \cup$
        $\{\mathsf{A} \in V \mid$ es gibt $\mathsf{A} \to \mathsf{BC} \in P$ mit
          $\mathsf{B} \in V[i, k]$ und $\mathsf{C} \in V[k+1, j]\}$

return $\mathsf{S} \in V[1, n]$ **?** „ja“ **:** „nein“



Der CYK-Algorithmus

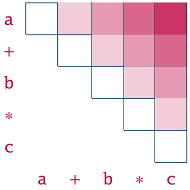
Eingabe: Wort $w = \mathtt{a}_1 \cdots \mathtt{a}_n$,
CNF-Grammatik $G = \langle V, \Sigma, P, \mathsf{S} \rangle$
Ausgabe: „ja“ wenn $w \in \mathbf{L}(G)$; sonst „nein“

for $i = 1, \ldots, n$: // Teilwörter der Länge 1
  $V[i,i] := \{\mathsf{A} \in V \mid \mathsf{A} \to \mathtt{a}_i \in P\}$

for $d = 1, \ldots, n-1$: // Differenz Endind. – Startind.
  for $i = 1, \ldots, n-d$: // Startindex
    $j := i + d$
    $V[i,j] := \emptyset$
    for $k = i, \ldots, j-1$: // Trennindex
      $V[i,j] := V[i,j] \cup$
        $\{\mathsf{A} \in V \mid$ es gibt $\mathsf{A} \to \mathsf{BC} \in P$ mit
          $\mathsf{B} \in V[i,k]$ und $\mathsf{C} \in V[k+1,j]\}$

return $\mathsf{S} \in V[1,n]$ **?** „ja“ **:** „nein“

$\Longleftarrow$



Der CYK-Algorithmus

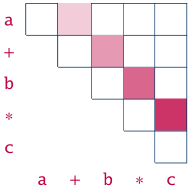
Eingabe: Wort $w = \mathtt{a}_1 \cdots \mathtt{a}_n$,
CNF-Grammatik $G = \langle V, \Sigma, P, \mathsf{S} \rangle$
Ausgabe: „ja“ wenn $w \in \mathbf{L}(G)$; sonst „nein“

for $i = 1, \ldots, n$: // Teilwörter der Länge 1
  $V[i,i] := \{\mathsf{A} \in V \mid \mathsf{A} \to \mathtt{a}_i \in P\}$

for $d = 1, \ldots, n-1$: // Differenz Endind. – Startind.
  for $i = 1, \ldots, n-d$: // Startindex
    $j := i + d$
    $V[i,j] := \emptyset$
    for $k = i, \ldots, j-1$: // Trennindex
      $V[i,j] := V[i,j] \cup$
        $\{\mathsf{A} \in V \mid$ es gibt $\mathsf{A} \to \mathsf{BC} \in P$ mit
          $\mathsf{B} \in V[i,k]$ und $\mathsf{C} \in V[k+1,j]\}$

return $\mathsf{S} \in V[1,n]$ **?** „ja“ **:** „nein“

$\Longleftarrow$



Der CYK-Algorithmus

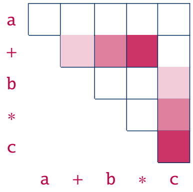
Eingabe: Wort $w = \mathtt{a}_1 \cdots \mathtt{a}_n$,
CNF-Grammatik $G = \langle V, \Sigma, P, \mathsf{S} \rangle$
Ausgabe: „ja“ wenn $w \in \mathbf{L}(G)$; sonst „nein“

for $i = 1, \ldots, n$: // Teilwörter der Länge 1
  $V[i,i] := \{\mathsf{A} \in V \mid \mathsf{A} \to \mathtt{a}_i \in P\}$

for $d = 1, \ldots, n-1$: // Differenz Endind. – Startind.
  for $i = 1, \ldots, n-d$: // Startindex
    $j := i + d$
    $V[i,j] := \emptyset$
    for $k = i, \ldots, j-1$: // Trennindex
      $V[i,j] := V[i,j] \cup$
        $\{\mathsf{A} \in V \mid$ es gibt $\mathsf{A} \to \mathsf{BC} \in P$ mit
          $\mathsf{B} \in V[i,k]$ und $\mathsf{C} \in V[k+1,j]\}$

return $\mathsf{S} \in V[1,n]$ **?** „ja“ **:** „nein“

$\Longleftarrow$



Beispiel 2

Grammatik:

S → CD | a   D → SE   C → AB
E → BA   A → a   B → b

Wort:

w = ababababa

	a	b	a	b	a	b	a	b	a
a	S,A	C	D	–	S	–	D	–	S
b		B	E	–	–	–	–	–	–
a			S,A	C	D	–	S	–	D
b				B	E	–	–	–	–
a					S, A	C	D	–	S
b						B	E	–	–
a							S, A	C	D
b								B	E
a									S,A

$S \in V[1,9] \rightsquigarrow w$ kann erzeugt werden

Das Pumping-Lemma für kontextfreie Sprachen

Rückblick: Pumpen regulärer Sprachen

Satz (Pumping Lemma): Für jede reguläre Sprache **L**
gibt es eine Zahl $n \geq 0$, so dass gilt:
für jedes Wort $z \in \mathbf{L}$ mit $|z| \geq n$
gibt es eine Zerlegung $z = uvw$ mit $|v| \geq 1$ und $|uv| \leq n$, so dass:
für jede Zahl $k \geq 0$ gilt: $uv^kw \in \mathbf{L}$

In Worten: Jedes ausreichend lange Wort einer regulären Sprache enthält ein Teilwort, welches beliebig oft wiederholt werden kann um weitere Wörter der Sprache zu erhalten.

Aber: Für viele kontextfreie Sprachen wie $\mathtt{a}^n\mathtt{b}^n$ trifft das nicht zu.

Pumpen für kontextfreie Sprachen

Satz (Pumping Lemma): Für jede kontextfreie Sprache **L**
gibt es eine Zahl $n \geq 0$, so dass gilt:
für jedes Wort $z \in \mathbf{L}$ mit $|z| \geq n$
gibt es eine Zerlegung $z = uvwxy$ mit $|vx| \geq 1$ und $|vwx| \leq n$, so dass:
für jede Zahl $k \geq 0$ gilt: $uv^kwx^ky \in \mathbf{L}$

Beispiel: Für die Sprache $\{\mathtt{a}^i\mathtt{b}^i \mid i \geq 0\}$ gilt der Satz. Wir wählen $n = 2$. Sei $z = \mathtt{a}^i\mathtt{b}^i$ mit $i \geq 1$ ein beliebiges Wort mit $|z| \geq 2$. Wir wählen die Zerlegung $u = \mathtt{a}^{i-1}$, $v = \mathtt{a}$, $w = \epsilon$, $x = \mathtt{b}$ und $y = \mathtt{b}^{i-1}$.
Dann ist $uv^kwx^ky = \mathtt{a}^{i-1}\mathtt{a}^k\mathtt{b}^k\mathtt{b}^{i-1} = \mathtt{a}^{i+k-1}\mathtt{b}^{i+k-1} \in \mathbf{L}$ für alle $k \geq 0$.

Kontextfreies Pumpen: Idee

Grundidee beim regulären Pumping Lemma:

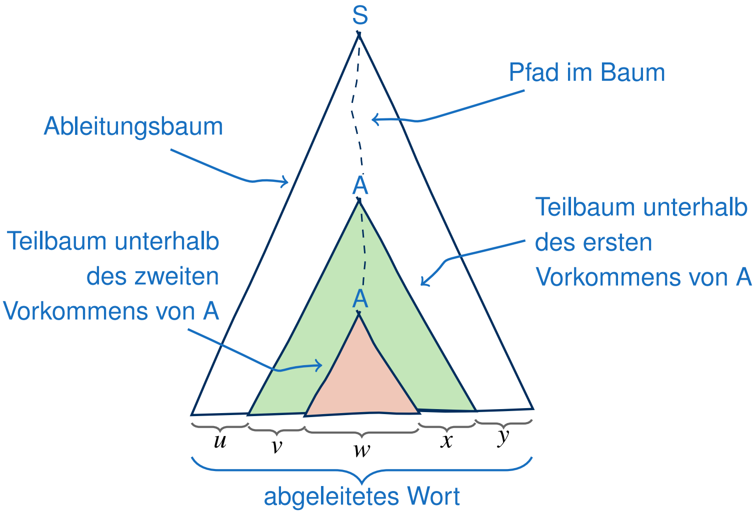
- Endliche Automaten haben nur endlich viele Zustände
- Also muss beim Akzeptieren langer Wörter ein Zustand mehrfach besucht werden:
$$q_0 \xrightarrow{u_1} \ldots \xrightarrow{u_{|u|}} p \xrightarrow{v_1} \ldots \xrightarrow{v_{|v|}} p \xrightarrow{w_1} \ldots \xrightarrow{w_{|w|}} q_n$$
- Also gibt es eine Schleife, die man beliebig oft durchlaufen kann

Grundidee beim kontextfreien Pumping Lemma:

- Grammatiken haben nur endlich viele Variablen
- Also muss beim Generieren langer Wörter eine Variable zu etwas expandiert werden, das diese Variable nochmal enthält:
$$\mathsf{S} \Rightarrow \ldots \Rightarrow u\,\underline{\mathsf{A}}\,y \Rightarrow u\,\underline{z}\,y \Rightarrow \ldots \Rightarrow u\,\underline{v\mathsf{A}x}\,y \Rightarrow \ldots \Rightarrow u\,\underline{vwx}\,y$$
- Also gibt es eine Schleife, die man beliebig oft durchlaufen kann

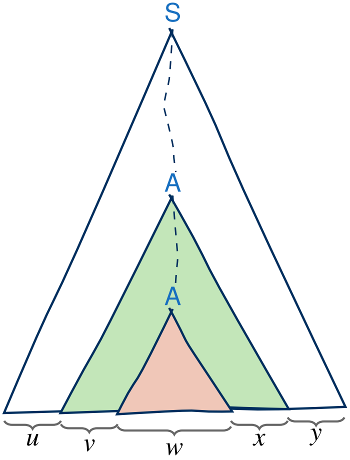
Ableitungsbäume aufpumpen

Die Idee des Lemmas lässt sich gut am Ableitungsbaum darstellen:



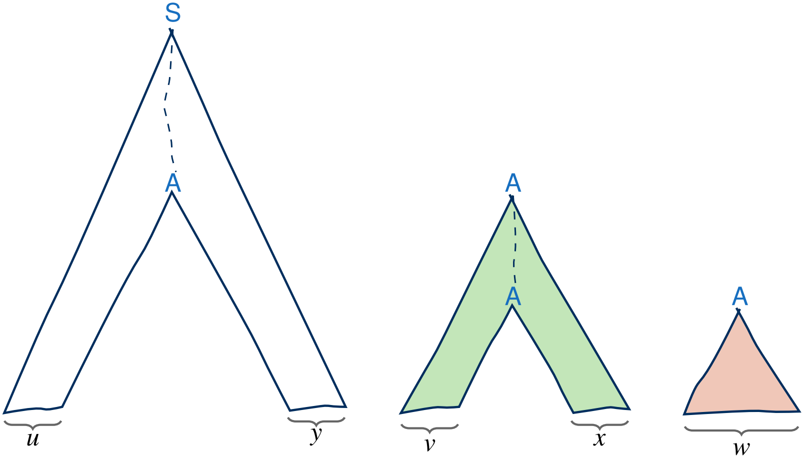
Ableitungsbäume aufpumpen (2)

Die Idee des Lemmas lässt sich gut am Ableitungsbaum darstellen:



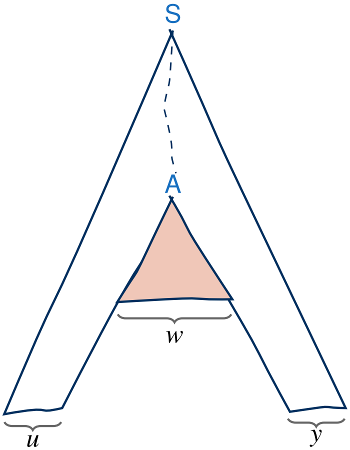
Ableitungsbäume aufpumpen (3)

Die Idee des Lemmas lässt sich gut am Ableitungsbaum darstellen:



Ableitungsbäume aufpumpen (4)

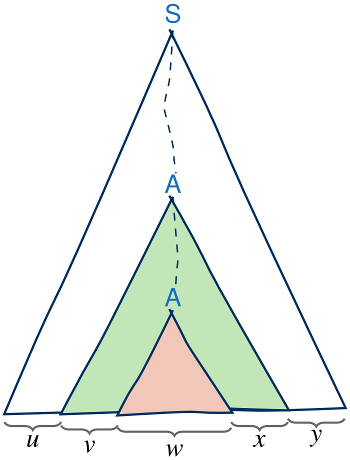
Die Idee des Lemmas lässt sich gut am Ableitungsbaum darstellen:



Abgeleitetes Wort:
uwy

Ableitungsbäume aufpumpen (5)

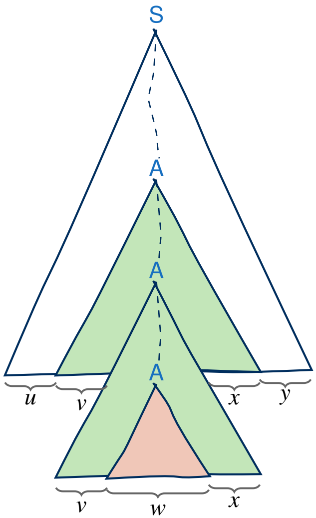
Die Idee des Lemmas lässt sich gut am Ableitungsbaum darstellen:



Abgeleitetes Wort:
$uvwxy$

Ableitungsbäume aufpumpen (6)

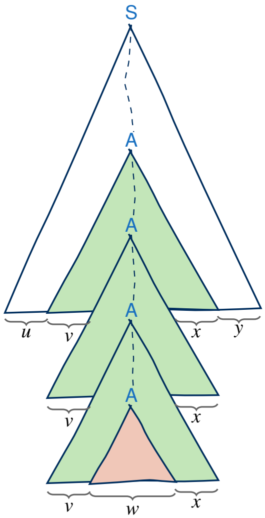
Die Idee des Lemmas lässt sich gut am Ableitungsbaum darstellen:



Abgeleitetes Wort:
$uvvwxxy$

Ableitungsbäume aufpumpen (7)

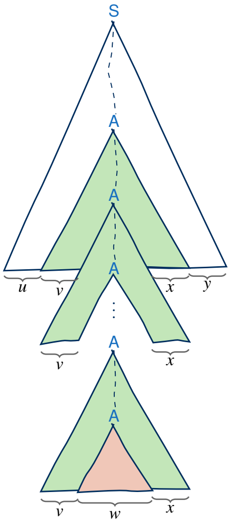
Die Idee des Lemmas lässt sich gut am Ableitungsbaum darstellen:



Abgeleitetes Wort:
$uvvvwxxxy$

Ableitungsbäume aufpumpen (4+k)

Die Idee des Lemmas lässt sich gut am Ableitungsbaum darstellen:



Abgeleitetes Wort:
uv^kwx^ky

Beweis des kontextfreien Pumping Lemma

Satz (Pumping Lemma): Für jede kontextfreie Sprache **L**
gibt es eine Zahl $n \geq 0$, so dass gilt:
für jedes Wort $z \in \mathbf{L}$ mit $|z| \geq n$
gibt es eine Zerlegung $z = uvwxy$ mit $|vx| \geq 1$ und $|vwx| \leq n$, so dass:
für jede Zahl $k \geq 0$ gilt: $uv^kwx^ky \in \mathbf{L}$

Beweis: Wir haben gesehen: wenn es einen Ableitungsbaum für z gibt, der einen Pfad enthält, in dem eine Variable A mehrfach vorkommt, dann gibt es eine Zerlegung $z = uvwxy$, die wesentliche Eigenschaften des Lemmas erfüllt.

Offene Fragen:

- Wieso sollte $|vx| \geq 1$ gelten?
- Wie groß muss n sein, damit der Ableitungsbaum garantiert solch einen Pfad mit doppelter Variable enthält?
- Weshalb gilt $|vwx| \leq n$?

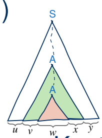
$\leadsto$ Hier hilft es, eine Grammatik in CNF anzunehmen

Beweis des kontextfreien Pumping Lemma (2)

Satz (Pumping Lemma): Für jede kontextfreie Sprache **L**
gibt es eine Zahl $n \geq 0$, so dass gilt:
für jedes Wort $z \in \mathbf{L}$ mit $|z| \geq n$
gibt es eine Zerlegung $z = uvwxy$ mit $|vx| \geq 1$ und $|vwx| \leq n$, so dass:
für jede Zahl $k \geq 0$ gilt: $uv^kwx^ky \in \mathbf{L}$

Beweis: Sei G eine Grammatik für die Sprache $\mathbf{L} \setminus \{\epsilon\}$ in CNF. (Diese existiert immer. Das Wort ϵ ist im Beweis nicht relevant, da sicherlich $n > 0$.)

Wieso sollte $|vx| \geq 1$ gelten?

Wenn man einen Ableitungsbaum in G findet, der die Form 

hat, dann muss der obere A-Knoten genau zwei Kinder haben (wie jeder innere Knoten in CNF). Da CNF-Grammatiken ϵ-frei sind, führt jeder Kindknoten zu einem nichtleeren Teilwort. Daher muss entweder v oder x mindestens ein Symbol enthalten, d.h. $|vx| \geq 1$.

Beweis des kontextfreien Pumping Lemma (3)

Beweis: Sei $G = \langle V, \Sigma, P, \mathsf{S} \rangle$ eine Grammatik für $\mathbf{L} \setminus \{\epsilon\}$ in CNF.

Wie groß muss n sein, damit der Ableitungsbaum garantiert solch einen Pfad mit doppelter Variable enthält?

Jeder Ableitungsbaum in G ist ein Binärbaum (bis auf die letzte Ebene, wo Variablen durch Terminale ersetzt werden).

- Ein Ableitungsbaum für z hat $|z|$ Blätter
- Ein Binärbaum mit $|z|$ Blättern muss Pfade der Länge $\geq \log_2 |z|$ enthalten[1]
  (Der größte Binärbaum mit Pfaden der Länge ℓ ist der, in dem alle Pfade diese Länge haben und der 2^ℓ Blätter hat)
- Jeder Pfad der Länge $\geq |V| - 1$ muss mindestens eine Variable doppelt enthalten
  (Schubfachprinzip)

$\leadsto$ Wenn $|z| > 2^{|V|-1}$ ist, dann gibt es einen Pfad, in dem eine Variable doppelt vorkommt, also funktioniert dafür jedes $n \geq 2^{|V|-1} + 1$

[1] Wir messen die Pfadlänge hier als Zahl der Elter-zu-Kind-Schritte. Ein Pfad der Länge ℓ hat also $\ell + 1$ Knoten.

Beweis des kontextfreien Pumping Lemma (4)

Satz (Pumping Lemma): Für jede kontextfreie Sprache **L**
gibt es eine Zahl $n \geq 0$, so dass gilt:
für jedes Wort $z \in \mathbf{L}$ mit $|z| \geq n$
gibt es eine Zerlegung $z = uvwxy$ mit $|vx| \geq 1$ und $|vwx| \leq n$, so dass:
für jede Zahl $k \geq 0$ gilt: $uv^kwx^ky \in \mathbf{L}$

Beweis: Gerade gezeigt: Ab Wortlänge $n \geq 2^{|V|-1} + 1$ gibt es Pfade mit Variablendopplungen.

Weshalb gilt $|vwx| \leq n$?

Auch hier hilft es, dass Ableitungsbäume binär sind:

- Ein Binärbaum der maximalen Pfadlänge ℓ hat maximal 2^ℓ Blätter
- Wir können annehmen, dass das obere doppelte Vorkommen der gewählten Variable maximal $|V|$ Schritte von der vorletzten Ebene (d.h. der letzten im inneren Binärbaum) entfernt ist
- Also kann der Baum unterhalb dieses Vorkommens maximal $2^{|V|}$ Blätter haben (dies sind die Symbole in vwx)

$\leadsto$ wir können $n = 2^{|V|}$ als konkreten Wert wählen (insbesondere ist $2^{|V|} \geq 2^{|V|-1} + 1$)

Beweis des kontextfreien Pumping Lemma (5)

Satz (Pumping Lemma): Für jede kontextfreie Sprache **L**
gibt es eine Zahl $n \geq 0$, so dass gilt:
für jedes Wort $z \in \mathbf{L}$ mit $|z| \geq n$
gibt es eine Zerlegung $z = uvwxy$ mit $|vx| \geq 1$ und $|vwx| \leq n$, so dass:
für jede Zahl $k \geq 0$ gilt: $uv^kwx^ky \in \mathbf{L}$

Beweis (Zusammenfassung):

- Zum Aufpumpen genügt es, dass in einem Pfad eines Ableitungsbaumes eine Variable doppelt auftritt
- Bei einer CNF-Grammatik $G = \langle V, \Sigma, P, \mathsf{S}\rangle$ muss das Wort dafür $\geq n = 2^{|V|}$ Zeichen haben
- $|vx| \geq 1$ und $|vwx| \leq n$ ergeben sich, weil CNF-Ableitungsbäume binär sind

□

Anwendung des Pumping Lemma

Wie schon bei regulären Sprachen ist die Pump-Eigenschaft notwendig, aber nicht hinreichend für Kontextfreiheit

$\rightsquigarrow$ Hauptanwendung: Erkennen nichtkontextfreier Sprachen

Beispiel: Die Sprache $\mathbf{L} = \{\mathtt{a}^i\mathtt{b}^i\mathtt{c}^i \mid i \geq 0\}$ ist nicht kontextfrei.

Beweis durch Widerspruch: angenommen, $\mathbf{L}$ wäre kontextfrei

- Dann gibt es eine Konstante n wie im Pumping Lemma.
- Dann muss es auch für das Wort $z = \mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$ eine geeignete Zerlegung $z = uvwxy$ geben.
- Wegen $|vwx| \leq n$ muss vwx entweder ein Teilwort von $\mathtt{a}^n\mathtt{b}^n$ oder von $\mathtt{b}^n\mathtt{c}^n$ sein.
- Fall 1: vwx ist Teilwort von $\mathtt{a}^n\mathtt{b}^n$. Weil $|vx| \geq 1$ und $vx \subseteq \{\mathtt{a}, \mathtt{b}\}^*$ hat uv^2wx^2y mehr $\mathtt{a}$ oder $\mathtt{b}$ als $\mathtt{c}$. Also ist $uv^2wx^2y \notin \mathbf{L}$.
- Fall 2: vwx ist Teilwort von $\mathtt{b}^n\mathtt{c}^n$. Analog zu Fall 1. $\square$

Zusammenfassung und Ausblick

Das Pumping-Lemma für kontextfreie Sprachen beruht auf der Ausnutzung von Schleifen in Ableitungen einer CNF-Grammatik

Die Sprache $\{\mathtt{a}^i\mathtt{b}^i\mathtt{c}^i \mid i \geq 0\}$ ist nicht kontextfrei

Offene Fragen:

- Wie steht es mit Abschlusseigenschaften bei kontextfreien Sprachen?
- Haben kontextfreie Sprachen ein Berechnungsmodell?
- Welche Probleme auf kontextfreien Grammatiken kann man lösen?