

FORMALE SYSTEME

13. Vorlesung: Das Pumping Lemma kontextfreier Sprachen

Markus Krötzsch
Lehrstuhl Wissensbasierte Systeme

TU Dresden, 23. November 2017

CYK: Praktische Umsetzung

Notation: Für $w = a_1 \cdots a_n$ schreiben wir $w_{i,j}$ für das Teilwort $a_i \cdots a_j$, 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 $A \rightarrow BC$ 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

CYK: Grundidee

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

Wie kann man prüfen, ob ein Wort $w = a_1 \cdots a_n$ 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 $S \rightarrow w$ in G gibt
- Falls $|w| > 1$, dann ist:
 $w \in \mathbf{L}(G)$ genau dann wenn es eine Regel $S \rightarrow AB$ und eine Zahl i gibt, so dass gilt

$$A \Rightarrow^* a_1 \cdots a_i \quad \text{und} \quad B \Rightarrow^* a_{i+1} \cdots a_n$$

Idee: Fall 2 reduziert das Problem $S \stackrel{?}{\Rightarrow}^* w$ in zwei einfachere Probleme $A \stackrel{?}{\Rightarrow}^* a_1 \cdots a_i$ und $B \stackrel{?}{\Rightarrow}^* a_{i+1} \cdots a_n$, die man allerdings für alle Regeln $S \rightarrow AB$ und Indizes i lösen muss

Notation

Für die Teilwörter $w_{i,j}$ gilt $i \leq j$

→ Die Mengen $V[i, j]$ können als Dreiecksmatrix notiert werden

Beispiel: Wir betrachten das Wort $w = a + b * c$ 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

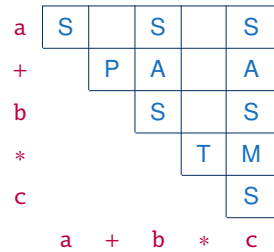
Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

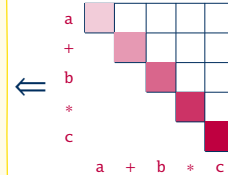
Berechnung der Mengen $V[i, j]$:



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

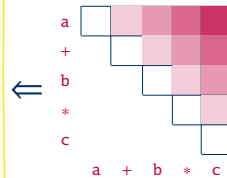
Der CYK-Algorithmus

Eingabe: Wort $w = a_1 \dots a_n$,
CNF-Grammatik $G = \langle V, \Sigma, P, S \rangle$
Ausgabe: „ja“ wenn $w \in L(G)$; sonst „nein“
for $i = 1, \dots, n$: // Teilwörter der Länge 1
 $V[i, i] := \{A \in V \mid A \rightarrow a_i \in P\}$
for $d = 1, \dots, n - 1$: // Differenz Endind. – Startind.
for $i = 1, \dots, n - d$: // Startindex
 $j := i + d$
 $V[i, j] := \emptyset$
for $k = i, \dots, j - 1$: // Trennindex
 $V[i, j] := V[i, j] \cup \{A \in V \mid \text{es gibt } A \rightarrow BC \in P \text{ mit } B \in V[i, k] \text{ und } C \in V[k + 1, j]\}$
return $S \in V[1, n]$? „ja“ : „nein“



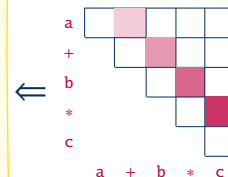
Der CYK-Algorithmus

Eingabe: Wort $w = a_1 \dots a_n$,
CNF-Grammatik $G = \langle V, \Sigma, P, S \rangle$
Ausgabe: „ja“ wenn $w \in L(G)$; sonst „nein“
for $i = 1, \dots, n$: // Teilwörter der Länge 1
 $V[i, i] := \{A \in V \mid A \rightarrow a_i \in P\}$
for $d = 1, \dots, n - 1$: // Differenz Endind. – Startind.
for $i = 1, \dots, n - d$: // Startindex
 $j := i + d$
 $V[i, j] := \emptyset$
for $k = i, \dots, j - 1$: // Trennindex
 $V[i, j] := V[i, j] \cup \{A \in V \mid \text{es gibt } A \rightarrow BC \in P \text{ mit } B \in V[i, k] \text{ und } C \in V[k + 1, j]\}$
return $S \in V[1, n]$? „ja“ : „nein“



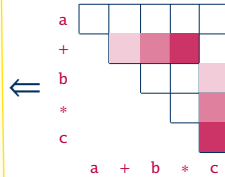
Der CYK-Algorithmus

Eingabe: Wort $w = a_1 \dots a_n$,
CNF-Grammatik $G = \langle V, \Sigma, P, S \rangle$
Ausgabe: „ja“ wenn $w \in L(G)$; sonst „nein“
for $i = 1, \dots, n$: // Teilwörter der Länge 1
 $V[i, i] := \{A \in V \mid A \rightarrow a_i \in P\}$
for $d = 1, \dots, n - 1$: // Differenz Endind. – Startind.
for $i = 1, \dots, n - d$: // Startindex
 $j := i + d$
 $V[i, j] := \emptyset$
for $k = i, \dots, j - 1$: // Trennindex
 $V[i, j] := V[i, j] \cup \{A \in V \mid \text{es gibt } A \rightarrow BC \in P \text{ mit } B \in V[i, k] \text{ und } C \in V[k + 1, j]\}$
return $S \in V[1, n]$? „ja“ : „nein“



Der CYK-Algorithmus

Eingabe: Wort $w = a_1 \dots a_n$,
 CNF-Grammatik $G = \langle V, \Sigma, P, S \rangle$
Ausgabe: „ja“ wenn $w \in L(G)$; sonst „nein“
for $i = 1, \dots, n$: // Teilwörter der Länge 1
 $V[i, i] := \{A \in V \mid A \rightarrow a_i \in P\}$
for $d = 1, \dots, n - 1$: // Differenz Endind. – Startind.
for $i = 1, \dots, n - d$: // Startindex
 $j := i + d$
 $V[i, j] := \emptyset$
for $k = i, \dots, j - 1$: // Trennindex
 $V[i, j] := V[i, j] \cup$
 $\{A \in V \mid \text{es gibt } A \rightarrow BC \in P \text{ mit}$
 $B \in V[i, k] \text{ und } C \in V[k + 1, j]\}$
return $S \in V[1, n] ?$ „ja“ : „nein“



Beispiel 2

Grammatik:
 $S \rightarrow CD \mid a$ $D \rightarrow SE$ $C \rightarrow AB$
 $E \rightarrow BA$ $A \rightarrow a$ $B \rightarrow b$

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

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

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 $z \in L$ 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: $uv^k w \in L$

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 $a^n b^n$ trifft das nicht zu.

Das Pumping-Lemma für kontextfreie Sprachen

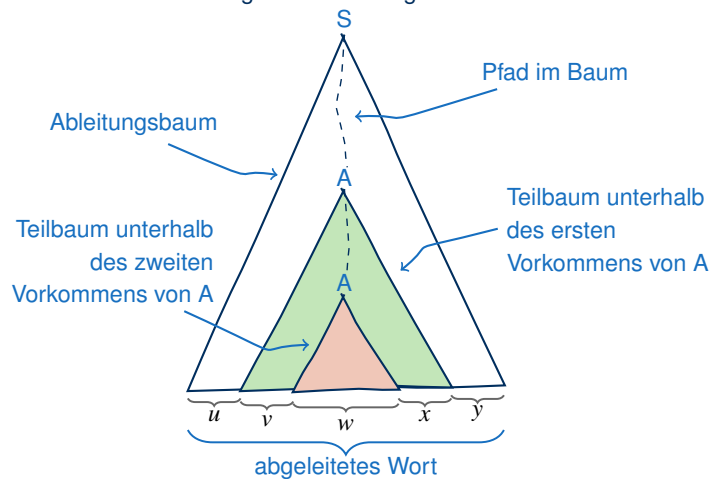
Pumpen für kontextfreie Sprachen

Satz (Pumping Lemma): Für jede kontextfreie Sprache L gibt es eine Zahl $n \geq 0$, so dass gilt:
 für jedes Wort $z \in L$ 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: $uv^kwx^ky \in L$

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

Ableitungsbäume aufpumpen

Die Idee des Lemmas lässt sich gut am Ableitungsbaum darstellen:



Kontextfreies Pumpen: Idee

Grundidee beim regulären Pumping Lemma:

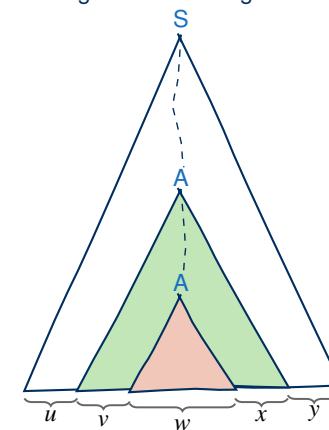
- Endliche Automaten haben nur endlich viele Zustände
 - Also muss beim Akzeptieren langer Wörter ein Zustand mehrfach besucht werden:
- $$q_0 \xrightarrow{u_1} \dots \xrightarrow{u_i} p \xrightarrow{v_1} \dots \xrightarrow{v_i} p \xrightarrow{w_1} \dots \xrightarrow{w_i} q_n$$
- Also gibt es eine Schleife, 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:
- $$S \Rightarrow \dots \Rightarrow uAy \Rightarrow uzzy \Rightarrow \dots \Rightarrow uvAxy \Rightarrow \dots \Rightarrow uvwxxy$$
- Also gibt es eine Schleife, die man beliebig oft durchlaufen kann

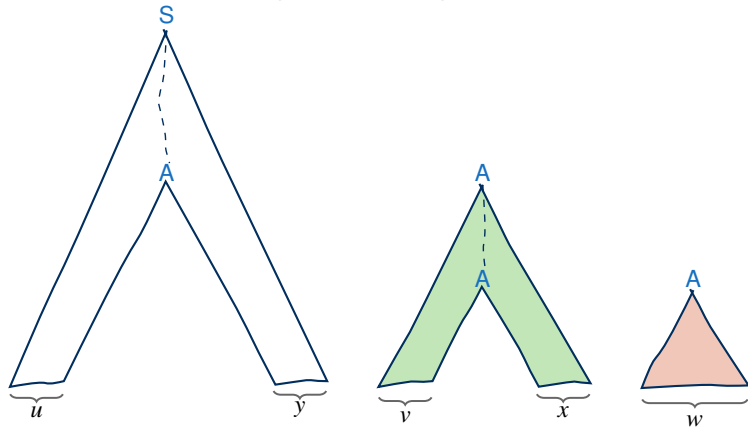
Ableitungsbäume aufpumpen (2)

Die Idee des Lemmas lässt sich gut am Ableitungsbaum darstellen:



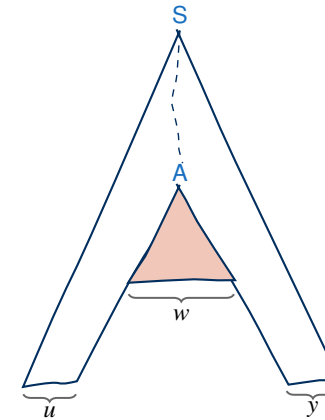
Ableitungsbäume aufpumpen (3)

Die Idee des Lemmas lässt sich gut am Ableitungsbaum darstellen:



Ableitungsbäume aufpumpen (4)

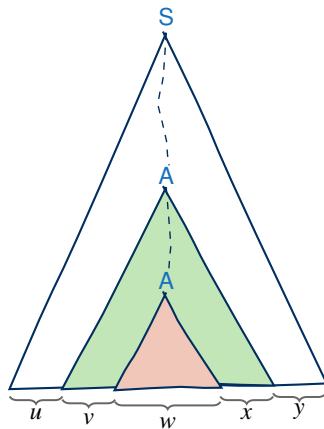
Die Idee des Lemmas lässt sich gut am Ableitungsbaum darstellen:



Abgeleitetes Wort:
 uwy

Ableitungsbäume aufpumpen (5)

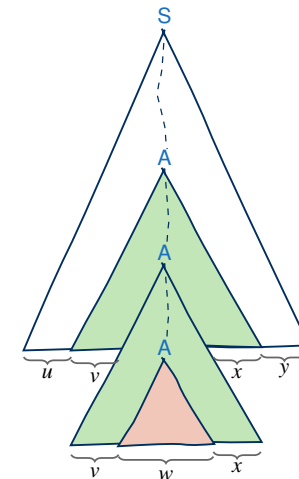
Die Idee des Lemmas lässt sich gut am Ableitungsbaum darstellen:



Abgeleitetes Wort:
 $uvwxy$

Ableitungsbäume aufpumpen (6)

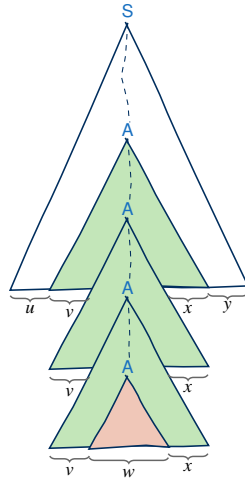
Die Idee des Lemmas lässt sich gut am Ableitungsbaum darstellen:



Abgeleitetes Wort:
 $uvwvwxxy$

Ableitungsbäume aufpumpen (7)

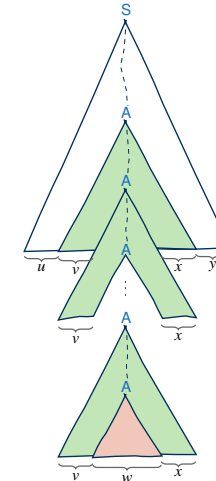
Die Idee des Lemmas lässt sich gut am Ableitungsbaum darstellen:



Abgeleitetes Wort:
 $uvvwxxy$

Ableitungsbäume aufpumpen (4+k)

Die Idee des Lemmas lässt sich gut am Ableitungsbaum darstellen:



Abgeleitetes Wort:
 $uv^kwx^k y$

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 $z \in L$ 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: $uv^kwx^ky \in L$

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 $|vwx| \leq n$?

→ 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 $z \in L$ 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: $uv^kwx^ky \in L$

Beweis: Sei G eine Grammatik für die Sprache $L \setminus \{\epsilon\}$ 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 $G = \langle V, \Sigma, P, S \rangle$ eine Grammatik für $L \setminus \{\epsilon\}$ 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
(Der größte Binärbaum mit Pfaden der Länge ℓ ist der, in dem alle Pfade diese Länge haben und der hat 2^ℓ Blätter)
- Jeder Pfad der Länge $\geq |V|$ muss mindestens eine Variable doppelt enthalten
(Schubfachprinzip)

→ Wenn $|z| \geq 2^{|V|}$ ist, dann gibt es einen Pfad, in dem eine Variable doppelt vorkommt

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 $z \in L$ 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: $uv^kwx^ky \in L$

Beweis: Gerade gezeigt: Die Konstante aus dem Lemma ist $n = 2^{|V|}$.

Weshalb gilt $|vwx| \leq n$?

Auch hier hilft es, dass Ableitungsbaume binär sind:

- Ein Binärbaum der Tiefe 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 $2^{|V|} = n$ Blätter haben
(dies sind die Symbole in vwx)

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 $z \in L$ 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: $uv^kwx^ky \in L$

Beweis (Zusammenfassung):

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

□

Anwendung des Pumping Lemma

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

→ Hauptanwendung: Erkennen nicht-kontextfreier Sprachen

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

Beweis durch Widerspruch: angenommen, L wäre kontextfrei

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

□

Zusammenfassung und Ausblick

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

Die Sprache $\{a^i b^i c^i \mid i \geq 0\}$ 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?