Kapitel 3

Kontextfreie Sprachen

3.3

Eigenschaften kontextfreier Sprachen

Prof. Dr. Robert Preis
Fachbereich Informatik
Fachhochschule Dortmund
Robert.Preis@fh-dortmund.de

Brennende Fragen für kontextfreie Sprachen

Frage 1: Ist die Sprache einer kontextfreien Grammatik leer oder wird mindestens ein Wort akzeptiert?

Antwort: Es wird mindestens ein Wort akzeptiert g.d.w. S erzeugend ist.

Frage 2: Ist ein Wort w in der Sprache einer kontextfreien Grammatik?

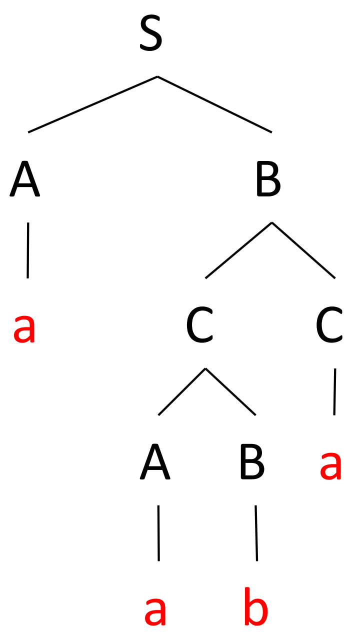
Frage 3: Sind zwei kontextfreie Grammatiken äquivalent, d.h. sind deren Sprachen identisch?

Frage 4: Kann man eine kontextfreie Grammatik kleiner machen, d.h. mit weniger Regeln?

Um diese Fragen zu beantworten, brauchen wir eine einfach zu analysierende Darstellung einer kontextfreien Grammatik: CNF!

Tests für kontextfreie Sprachen

Frage 2: Ist ein Wort w in der Sprache einer kontextfreien Grammatik?



1. Erzeuge Chomsky-Normalform
2. Dann wissen wir über ein Wort w der Länge |w| :
   - Es gibt genau |w| Blätter.
   - Die Variablenknoten bilden einen Binärbaum (d.h. immer entweder 0 oder 2 Kinder als Variablen).
   - Es gibt genau 2|w|−1 Variablenknoten.

→ ***Jede Ableitung für w hat die Länge 2|w|-1***

Also funktioniert dieser Test:

Generiere alle Bäume mit 2|w|−1 Variablenknoten und teste, ob ein Baum w erzeugt. Geht, ist aber sehr ineffizient, da exponentiell viele Bäume möglich sind …

…es geht besser:

Wortproblem: Motivation

P = { S → AB | BC, A → BA | a, B → CC | b, C → AB | a }

Ist w = baaba ∈ L(G) ? D.h. gibt es S → baaba ?*

Es gibt nur die folgenden Start-Möglichkeiten:

1. S → AB mit
   a) A →* b und B →* aaba *aber gibt es A →* b und B →* aaba ?*
   b) A →* ba und B →* aba *aber gibt es A →* ba und B →* aba ?*
   c) A →* baa und B →* ba *aber gibt es A →* baa und B →* ba ?*
   d) A →* baab und B →* a *aber gibt es A →* baab und B →* a ?*
2. S → BC mit
   a) B →* b und C →* aaba *aber gibt es B →* b und C →* aaba ?*
   b) B →* ba und C →* aba *aber gibt es B →* ba und C →* aba ?*
   c) B →* baa und C →* ba *aber gibt es B →* baa und C →* ba ?*
   d) B →* baab und C →* a *aber gibt es B →* baab und C →* a ?*

Für ein Wort der Länge 5 brauche ich Antworten der Länge <=4.

Es kann viele Doppelte geben. Deshalb muss ich es von klein an aufbauen!

Der Cocke-Younger-Kasami (CYK) Algorithmus: Sockel der Pyramide

Eingabe: Grammatik G = (V,T,P,S) in Chomsky-NF und $w=w_1 \ldots w_n \in T^*$

Idee: Berechne die Mengen

$$V_{I,J} = \{A \in V \mid A \to^* w_I \ldots w_J\},$$

d.h. alle Variablen, die das Wort vom I-ten bis J-ten Buchstaben erzeugen.

Baue den Sockel einer Pyramide (Beispiel n=5):

1. Sockel: Das Wort w
2. Unterste Ebene (I=J): $V_{I,I} = \{A \in V \mid A \to w_I \in P\}$
   (hierfür braucht man nur die Regeln mit einem Terminal zu betrachten)

$V_{1,1}$	$V_{2,2}$	$V_{3,3}$	$V_{4,4}$	$V_{5,5}$
w_1	w_2	w_3	w_4	w_5

$P = \{ \quad S \to AB \mid BC,$
$A \to BA \mid a,$
$B \to CC \mid b,$
$C \to AB \mid a \}$

{B}	**{A,C}**	**{A,C}**	**{B}**	**{A,C}**
b	**a**	**a**	**b**	**a**

Ist w = baaba ∈ L(G) ?

Der Cocke-Younger-Kasami (CYK) Algorithmus: Neue Ebene

Baue die nächste Ebene der Pyramide auf:

3. $V_{l,l+1} = \{ A \in V \mid \exists\, A \rightarrow BC \in P$ mit $B \in V_{l,l} \wedge C \in V_{l+1,l+1} \}$
   d.h. wenn B das Teilwort w_l und C das Teilwort w_{l+1} erzeugt und es eine Regel $A \rightarrow BC$ gibt, dann erzeugt A das Teilwort $w_l w_{l+1}$.

$\boxed{V_{1,2}}$	$V_{2,3}$	$V_{3,4}$	$V_{4,5}$	
$V_{1,1}$	$V_{2,2}$	$V_{3,3}$	$V_{4,4}$	$V_{5,5}$
w_1	w_2	w_3	w_4	w_5

$P = \{ \quad S \rightarrow AB \mid BC,$
$\quad A \rightarrow BA \mid a,$
$\quad B \rightarrow CC \mid b,$
$\quad C \rightarrow AB \mid a \}$

Ist $w = baaba \in L(G)$?

$\boxed{\{S,A\}}$	{B}	{S,C}	{S,A}	
{B}	{A,C}	{A,C}	{B}	{A,C}
b	a	a	b	a

D.h. es gibt z.B. $S \rightarrow^* ba$ und $A \rightarrow^* ba$

Der Cocke-Younger-Kasami (CYK) Algorithmus: Weitere Ebenen

Baue weitere Ebene der Pyramide auf:

4. $V_{I,J} = \{ A \in V \mid \exists\, A \rightarrow BC \in P$ mit $B \in V_{I,K} \wedge C \in V_{K+1,J}$ für ein $I \leq K < J \}$
   d.h. wenn B das Teilwort $w_I \ldots w_K$ und
   C das Teilwort $w_{K+1} \ldots w_J$ erzeugt und
   es eine Regel $A \rightarrow BC$ gibt,
   dann erzeugt A das Teilwort $w_I \ldots w_J$.

$\boxed{V_{1,3}}$	$V_{2,4}$	$V_{3,5}$		
$V_{1,2}$	$V_{2,3}$	$V_{3,4}$	$V_{4,5}$	
$V_{1,1}$	$V_{2,2}$	$V_{3,3}$	$V_{4,4}$	$V_{5,5}$
w_1	w_2	w_3	w_4	w_5

P = { S → AB | BC,
A → BA | a,
B → CC | b,
C → AB | a }

Ist w = baaba ∈ L(G) ?

$\boxed{\{\}}$	{B}	{B}		
{S,A}	{B}	{S,C}	{S,A}	
{B}	{A,C}	{A,C}	{B}	{A,C}
b	a	a	b	a

D.h. es gibt z.B. kein S →* baa, aber B →* aab und B →* aba.

Der Cocke-Younger-Kasami (CYK) Algorithmus: Die Spitze

Baue die Pyramide bis zur Spitze auf:

5. $V_{I,J} = \{ A \in V \mid \exists\, A \rightarrow BC \in P$ mit $B \in V_{I,K} \wedge C \in V_{K+1,J}$ für ein $I \leq K < J \}$

   d.h. wenn B das Teilwort $w_I \dots w_K$ und C das Teilwort $w_{K+1} \dots w_J$ erzeugt und es eine Regel $A \rightarrow BC$ gibt, dann erzeugt A das Teilwort $w_I \dots w_J$.

6. Akzeptiere w genau dann, wenn $S \in V_{1,|w|}$

$\boxed{V_{1,5}}$

$V_{1,4}$ $V_{2,5}$

$V_{1,3}$ $V_{2,4}$ $V_{3,5}$

$V_{1,2}$ $V_{2,3}$ $V_{3,4}$ $V_{4,5}$

$V_{1,1}$ $V_{2,2}$ $V_{3,3}$ $V_{4,4}$ $V_{5,5}$

w_1 w_2 w_3 w_4 w_5

$P = \{ \quad S \rightarrow AB \mid BC,$
$\quad A \rightarrow BA \mid a,$
$\quad B \rightarrow CC \mid b,$
$\quad C \rightarrow AB \mid a \}$

Ist w = baaba $\in L(G)$?

$\boxed{\{S,A,C\}}$

$\{\}$ $\{S,A,C\}$

$\{\}$ $\{B\}$ $\{B\}$

$\{S,A\}$ $\{B\}$ $\{S,C\}$ $\{S,A\}$

$\{B\}$ $\{A,C\}$ $\{A,C\}$ $\{B\}$ $\{A,C\}$

b a a b a

Es ist $S \in V_{1,5}$, also $w \in L(G)$!

CYK Algorithmus: Ableitungsbaum

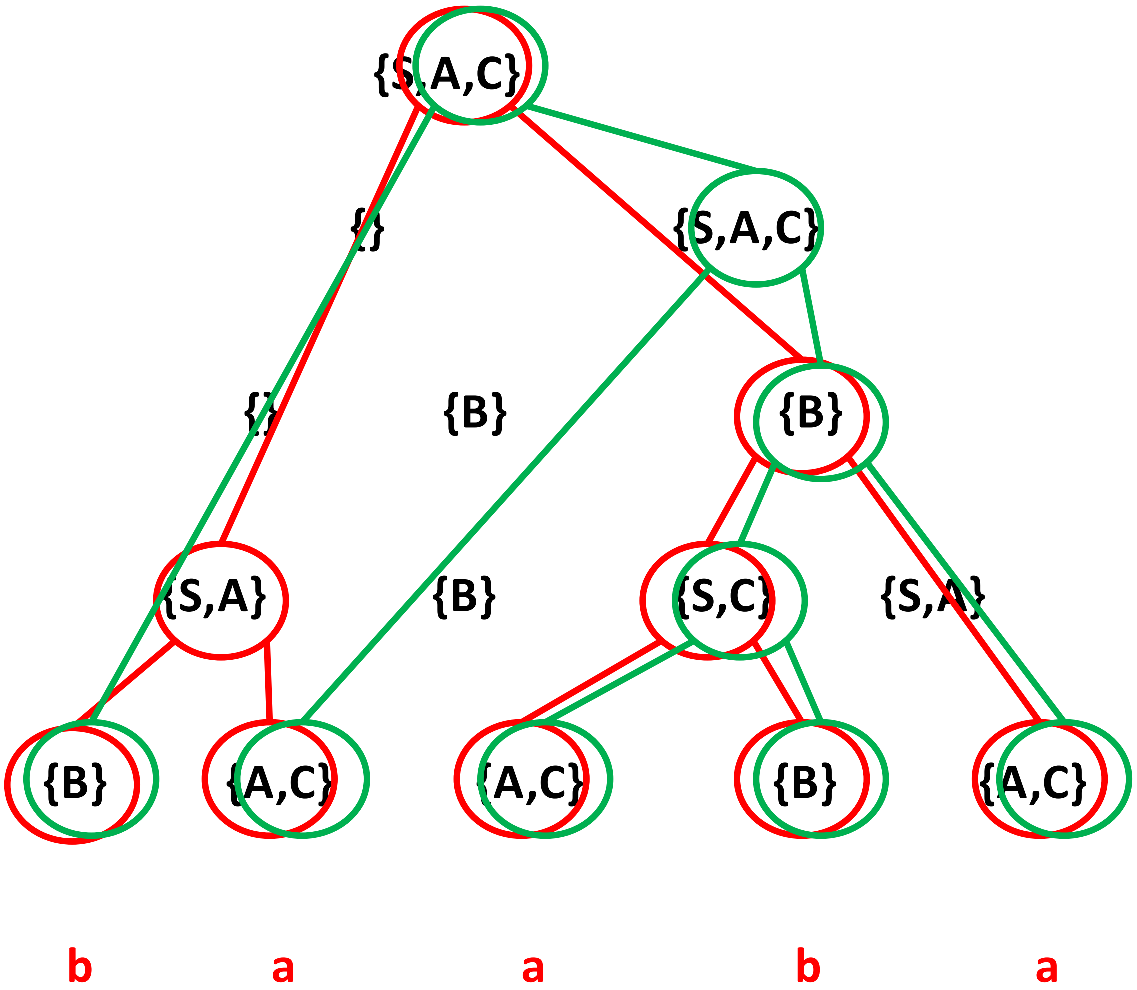
P = { S → AB | BC,
A → BA | a,
B → CC | b,
C → AB | a }

Ist w = baaba ∈ L(G) ?

Es ist $S \in V_{1,5}$, also w ∈ L(G) !

Aber wie sieht dann der Ableitungsbaum aus?

Um ihn zu rekonstruieren, hätte man sich merken sollen, wie die einzelnen Mengen zustande gekommen sind.



Zusammenfassung

- Man kann testen ob eine kontextfreie **Sprache leer** ist.
- **Wortproblem**: Man kann testen ob ein Wort zu einer kontextfreien Sprache gehört, Besonders leicht geht das, wenn die Grammatik in einer Chomsky-Normalform ist. Es gibt dafür effiziente Algorithmen.