

Kontextfreie Sprachen: Teil II

Prof. Dr. David Sabel

LFE Theoretische Informatik



- Effiziente Lösung des Wortproblems für CFLs:
Der Cocke-Younger-Kasami-Algorithmus
- Kellerautomaten
- Deterministisch kontextfreie Sprachen
- Entscheidbarkeitsresultate für kontextfreie Sprachen

Effizientes Lösen des Wortproblems für CFLs

- Algorithmus für Typ 1-Sprachen hat exponentielle Laufzeit
- Jetzt: Algorithmus von Cocke, Younger und Kasami für CFLs
- Veröffentlicht in den 1960er Jahren
- Kurz: CYK-Algorithmus
- Polynomielle Laufzeit

Idee des CYK-Algorithmus

- Eingabe:
 - CFG $G = (V, \Sigma, P, S)$ in **Chomsky-Normalform** und
 - Wort $w \in \Sigma^*$
- Ausgabe:
 - ja, wenn $w \in L(G)$,
 - nein, wenn $w \notin L(G)$
- Grundidee des Algorithmus:
(Rekursiver) Test, ob Variable A ein Wort u erzeugt.
- Verwende Test zum Prüfen, ob S das Wort w erzeugt.

Idee des CYK-Algorithmus (2)

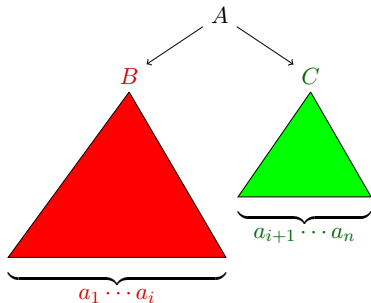
Prüfe, ob $A \in V$ ein Wort $u = a_1 \cdots a_n$ erzeugt:

- Wenn $u = a \in \Sigma$, dann prüfe ob $A \rightarrow a \in P$

Idee des CYK-Algorithmus (2)

Prüfe, ob $A \in V$ ein Wort $u = a_1 \cdots a_n$ erzeugt:

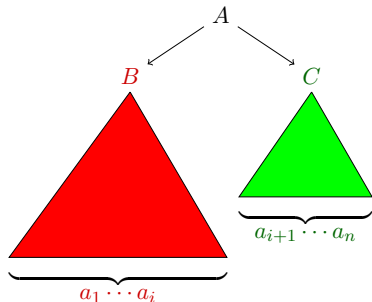
- Wenn $u = a \in \Sigma$, dann prüfe ob $A \rightarrow a \in P$
- Anderenfalls ($|u| > 1$) kann u nur erzeugt werden, wenn:
 - Es gibt Produktion $A \rightarrow BC \in P$
 - Es gibt Index $1 \leq i < n$, sodass:
 - B erzeugt $a_1 \cdots a_i$ und
 - C erzeugt $a_{i+1} \cdots a_n$



Idee des CYK-Algorithmus (2)

Prüfe, ob $A \in V$ ein Wort $u = a_1 \cdots a_n$ erzeugt:

- Wenn $u = a \in \Sigma$, dann prüfe ob $A \rightarrow a \in P$
- Anderenfalls ($|u| > 1$) kann u nur erzeugt werden, wenn:
 - Es gibt Produktion $A \rightarrow BC \in P$
 - Es gibt Index $1 \leq i < n$, sodass:
 - B erzeugt $a_1 \cdots a_i$ und
 - C erzeugt $a_{i+1} \cdots a_n$



- Daher prüfe für **alle** $A \rightarrow BC \in P$ und **alle** i mit $1 \leq i < n$ **rekursiv**, ob B das Wort $w = a_1 \cdots a_i$ und C das Wort $a_{i+1} \cdots a_n$ erzeugt.

Beispiel

$S \rightarrow AB \mid BA, \quad A \rightarrow AA \mid AB \mid a, \quad B \rightarrow BB \mid b$

S erzeugt $bbbaab$, denn $S \rightarrow BA$ und

- B erzeugt bbb , denn $B \rightarrow BB$ und
 - B erzeugt bb , denn $B \rightarrow BB$ und $B \rightarrow b$ und $B \rightarrow b$
 - B erzeugt b , denn $B \rightarrow b$
- A erzeugt aab , denn $A \rightarrow AB$ und
 - A erzeugt aa , denn $A \rightarrow AA$ und
 - A erzeugt a , denn $A \rightarrow a$ und
 - A erzeugt a , denn $A \rightarrow a$
 - B erzeugt b , denn $B \rightarrow b$

Beispiel

$S \rightarrow AB \mid BA, \quad A \rightarrow AA \mid AB \mid a, \quad B \rightarrow BB \mid b$

S erzeugt $bbbaab$, denn $S \rightarrow BA$ und

- B erzeugt bbb , denn $B \rightarrow BB$ und
 - B erzeugt bb , denn $B \rightarrow BB$ und $B \rightarrow b$ und $B \rightarrow b$
 - B erzeugt b , denn $B \rightarrow b$
- A erzeugt aab , denn $A \rightarrow AB$ und
 - A erzeugt aa , denn $A \rightarrow AA$ und
 - A erzeugt a , denn $A \rightarrow a$ und
 - A erzeugt a , denn $A \rightarrow a$
 - B erzeugt b , denn $B \rightarrow b$

Bevor der rekursive Algorithmus diesen richtigen Pfad findet, sucht er einige erfolglose ab, z.B. für $S \rightarrow AB$

- Prüfe, ob A erzeugt b und B erzeugt $bbab$ gilt.
- Prüfe, ob A erzeugt bb und B erzeugt bab gilt.
- Prüfe, ob A erzeugt bbb und B erzeugt ab gilt.
- Prüfe, ob A erzeugt $bbba$ und B erzeugt b gilt.

Naiv finden wiederholt dieselben Tests statt, z.B. ob A erzeugt b gilt.

Idee des CYK-Algorithmus (3)

- Effizienz: Statt Rekursion verwende **dynamische Programmierung**
- Algorithmus berechnet Menge $V(i, j) \subseteq V$, sodass

$$V(i, j) = \{A \in V \mid A \Rightarrow^* a_i \cdots a_{i+j-1}\}$$

„ $V(i, j)$ enthält alle $A \in V$, die $a_i \cdots a_{i+j-1}$ erzeugen“

- Berechnung der $V(i, j)$:
 - Starte mit $V(i, 1) = \{A \mid A \rightarrow a_i \in P\}$.
 - Berechne $V(i, j)$ mit ansteigender Länge j .

Für $j > 1$ gilt:

$A \in V(i, j)$ g.d.w.

$A \rightarrow BC \in P$ und $B \in V(i, k)$, $C \in V(i+k, j-k)$

Berechnung: für festes (i, j) betrachte alle k mit $k = 1, 2, \dots, j-1$

- Finaler Schritt: Prüfe, ob $S \in V(1, n)$ ist.

Algorithmus 8: CYK-Algorithmus

Eingabe: CFG $G = (V, \Sigma, P, S)$ in Chomsky-NF und Wort $w = a_1 \cdots a_n \in \Sigma^*$

Ausgabe: Ja, wenn $w \in L(G)$ und Nein, wenn $w \notin L(G)$

Beginn

für $i = 1$ bis n **tue**

└ $V(i, 1) = \{A \in V \mid A \rightarrow a_i \in P\}$

für $j = 2$ bis n **tue**

└ **für** $i = 1$ bis $n + 1 - j$ **tue**

└└ $V(i, j) = \emptyset;$

└└ **für** $k = 1$ bis $j - 1$ **tue**

└└└ $V(i, j) = V(i, j) \cup \left\{ A \in V \mid \begin{array}{l} A \rightarrow BC \in P, \\ B \in V(i, k), \\ C \in V(i + k, j - k) \end{array} \right\}$

└└└ **wenn** $S \in V(1, n)$ **dann**

└└└└ return Ja

└└└ **sonst**

└└└└ return Nein

Laufzeit des CYK-Algorithmus

- Drei geschachtelte Für-Schleifen
- Im inneren wird noch über alle Produktionen aus P iteriert
- Mit $n = |w|$ und $|P| = \text{Anzahl der Iterationen}$ kann die Laufzeitkomplexität mit $\mathcal{O}(n^3 \cdot |P|)$ abgeschätzt werden.

Theorem

Das Wortproblem für kontextfreie Sprachen kann in Polynomialzeit entschieden werden.

Beispiel

Sei $w = bbddc$ und $G = (\{S, A, B\}, \{b, c, d\}, P, S)$ mit
 $P = \{S \rightarrow AC, A \rightarrow BE, A \rightarrow BD, E \rightarrow AD, C \rightarrow c, B \rightarrow b, D \rightarrow d\}$

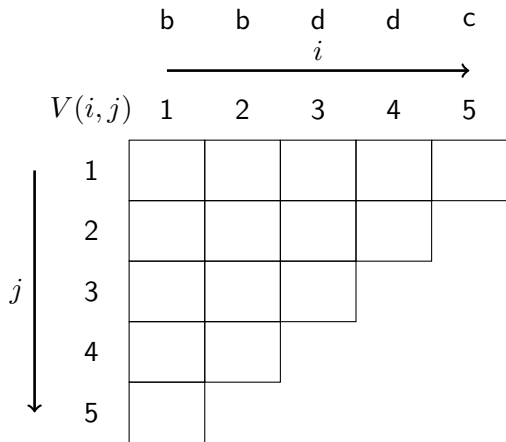
$V(i, j)$ -Tabelle ist zunächst leer:

		b	b	d	d	c
		i				
		→				
	$V(i, j)$	1	2	3	4	5
j	1					
	2					
	3					
	4					
	5					

Beispiel (2)

Sei $w = bbddc$ und $G = (\{S, A, B\}, \{b, c, d\}, P, S)$ mit
 $P = \{S \rightarrow AC, A \rightarrow BE, A \rightarrow BD, E \rightarrow AD, C \rightarrow c, B \rightarrow b, D \rightarrow d\}$

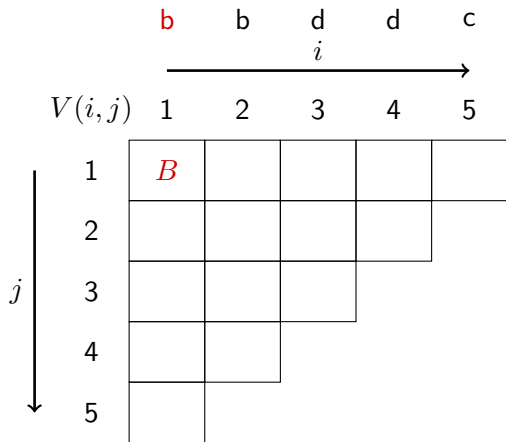
Füllen der $V(i, 1)$ -Einträge:



Beispiel (2)

Sei $w = bbddc$ und $G = (\{S, A, B\}, \{b, c, d\}, P, S)$ mit
 $P = \{S \rightarrow AC, A \rightarrow BE, A \rightarrow BD, E \rightarrow AD, C \rightarrow c, B \rightarrow b, D \rightarrow d\}$

Füllen der $V(i, 1)$ -Einträge:



Beispiel (2)

Sei $w = bbddc$ und $G = (\{S, A, B\}, \{b, c, d\}, P, S)$ mit
 $P = \{S \rightarrow AC, A \rightarrow BE, A \rightarrow BD, E \rightarrow AD, C \rightarrow c, B \rightarrow b, D \rightarrow d\}$

Füllen der $V(i, 1)$ -Einträge:

		b	b	d	d	c
		i				
		→				
	$V(i, j)$	1	2	3	4	5
j	1	B	B			
	2					
	3					
	4					
	5					

Beispiel (2)

Sei $w = bbddc$ und $G = (\{S, A, B\}, \{b, c, d\}, P, S)$ mit
 $P = \{S \rightarrow AC, A \rightarrow BE, A \rightarrow BD, E \rightarrow AD, C \rightarrow c, B \rightarrow b, D \rightarrow d\}$

Füllen der $V(i, 1)$ -Einträge:

		b	b	d	d	c
		i				
		→				
	$V(i, j)$	1	2	3	4	5
j	1	B	B	D		
	2					
	3					
	4					
	5					

Beispiel (2)

Sei $w = bbddc$ und $G = (\{S, A, B\}, \{b, c, d\}, P, S)$ mit
 $P = \{S \rightarrow AC, A \rightarrow BE, A \rightarrow BD, E \rightarrow AD, C \rightarrow c, B \rightarrow b, D \rightarrow d\}$

Füllen der $V(i, 1)$ -Einträge:

		b	b	d	d	c
		i				
		→				
	$V(i, j)$	1	2	3	4	5
j	1	B	B	D	D	
	2					
	3					
	4					
	5					

Beispiel (2)

Sei $w = bbddc$ und $G = (\{S, A, B\}, \{b, c, d\}, P, S)$ mit
 $P = \{S \rightarrow AC, A \rightarrow BE, A \rightarrow BD, E \rightarrow AD, C \rightarrow c, B \rightarrow b, D \rightarrow d\}$

Füllen der $V(i, 1)$ -Einträge:

		b	b	d	d	c
		i				
		→				
	$V(i, j)$	1	2	3	4	5
j	1	B	B	D	D	C
	2					
	3					
	4					
	5					

Beispiel (3)

Sei $w = bbddc$ und $G = (\{S, A, B\}, \{b, c, d\}, P, S)$ mit
 $P = \{S \rightarrow AC, A \rightarrow BE, A \rightarrow BD, E \rightarrow AD, C \rightarrow c, B \rightarrow b, D \rightarrow d\}$

		b	b	d	d	c
		\xrightarrow{i}				
	$V(i, j)$	1	2	3	4	5
$\downarrow j$	1	B	B	D	D	C
	2					
	3					
	4					
	5					

3-fach Schleife:

Beispiel (3)

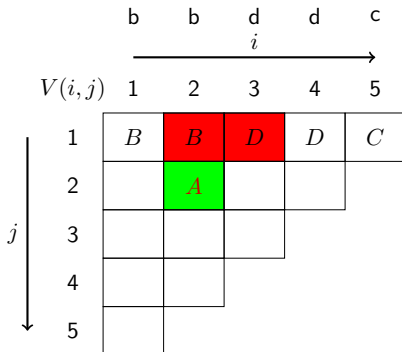
Sei $w = b b d d c$ und $G = (\{S, A, B\}, \{b, c, d\}, P, S)$ mit
 $P = \{S \rightarrow AC, A \rightarrow BE, A \rightarrow BD, E \rightarrow AD, C \rightarrow c, B \rightarrow b, D \rightarrow d\}$

		b	b	d	d	c
		\xrightarrow{i}				
	$V(i, j)$	1	2	3	4	5
$\downarrow j$	1	B	B	D	D	C
	2					
	3					
	4					
	5					

3-fach Schleife: $j=2, i=1, k=1: V(1, 2) = V(1, 2) \cup \left\{ A \left| \begin{array}{l} A \rightarrow BC, \\ B \in V(1, 1), \\ C \in V(2, 1) \end{array} \right. \right\}$

Beispiel (3)

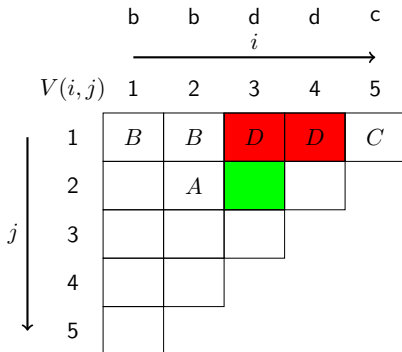
Sei $w = b b d d c$ und $G = (\{S, A, B\}, \{b, c, d\}, P, S)$ mit
 $P = \{S \rightarrow AC, A \rightarrow BE, A \rightarrow BD, E \rightarrow AD, C \rightarrow c, B \rightarrow b, D \rightarrow d\}$



3-fach Schleife: $j=2, i=2, k=1: V(2, 2) = V(2, 2) \cup \left\{ A \left| \begin{array}{l} A \rightarrow BC, \\ B \in V(2, 1), \\ C \in V(3, 1) \end{array} \right. \right\}$

Beispiel (3)

Sei $w = b b d d c$ und $G = (\{S, A, B\}, \{b, c, d\}, P, S)$ mit
 $P = \{S \rightarrow AC, A \rightarrow BE, A \rightarrow BD, E \rightarrow AD, C \rightarrow c, B \rightarrow b, D \rightarrow d\}$



3-fach Schleife: $j=2, i=3, k=1$: $V(3, 2) = V(3, 2) \cup \left\{ A \mid \begin{array}{l} A \rightarrow BC, \\ B \in V(3, 1), \\ C \in V(4, 1) \end{array} \right\}$

Beispiel (3)

Sei $w = b b d d c$ und $G = (\{S, A, B\}, \{b, c, d\}, P, S)$ mit
 $P = \{S \rightarrow AC, A \rightarrow BE, A \rightarrow BD, E \rightarrow AD, C \rightarrow c, B \rightarrow b, D \rightarrow d\}$

		b	b	d	d	c
		\xrightarrow{i}				
	$V(i, j)$	1	2	3	4	5
j	1	B	B	D	D	C
	2		A			
	3					
	4					
	5					

3-fach Schleife: $j=2, i=4, k=1: V(4, 2) = V(4, 2) \cup \left\{ A \left| \begin{array}{l} A \rightarrow BC, \\ B \in V(4, 1), \\ C \in V(5, 1) \end{array} \right. \right\}$

Beispiel (3)

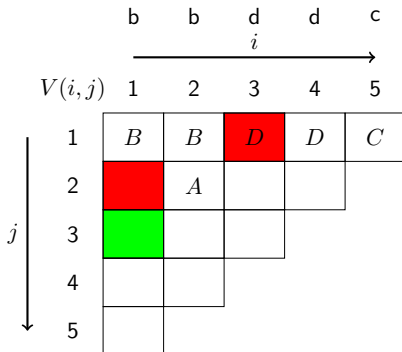
Sei $w = bdddc$ und $G = (\{S, A, B\}, \{b, c, d\}, P, S)$ mit
 $P = \{S \rightarrow AC, A \rightarrow BE, A \rightarrow BD, E \rightarrow AD, C \rightarrow c, B \rightarrow b, D \rightarrow d\}$

		b	b	d	d	c
		\xrightarrow{i}				
$V(i, j)$		1	2	3	4	5
$\downarrow j$	1	B	B	D	D	C
	2		A			
	3					
	4					
	5					

3-fach Schleife: $j=3, i=1, k=1: V(1, 3) = V(1, 3) \cup \left\{ A \left| \begin{array}{l} A \rightarrow BC, \\ B \in V(1, 1), \\ C \in V(2, 2) \end{array} \right. \right\}$

Beispiel (3)

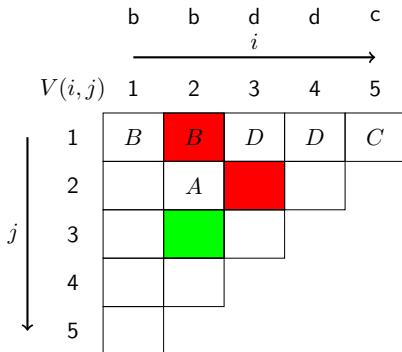
Sei $w = b b d d c$ und $G = (\{S, A, B\}, \{b, c, d\}, P, S)$ mit
 $P = \{S \rightarrow AC, A \rightarrow BE, A \rightarrow BD, E \rightarrow AD, C \rightarrow c, B \rightarrow b, D \rightarrow d\}$



3-fach Schleife: $j=3, i=1, k=2$: $V(1, 3) = V(1, 3) \cup \left\{ A \left| \begin{array}{l} A \rightarrow BC, \\ B \in V(1, 2), \\ C \in V(3, 1) \end{array} \right. \right\}$

Beispiel (3)

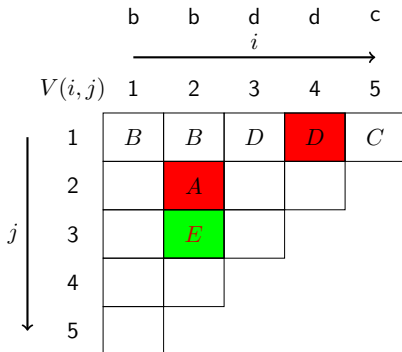
Sei $w = b b d d c$ und $G = (\{S, A, B\}, \{b, c, d\}, P, S)$ mit
 $P = \{S \rightarrow AC, A \rightarrow BE, A \rightarrow BD, E \rightarrow AD, C \rightarrow c, B \rightarrow b, D \rightarrow d\}$



3-fach Schleife: $j=3, i=2, k=1: V(2, 3) = V(2, 3) \cup \left\{ A \mid \begin{array}{l} A \rightarrow BC, \\ B \in V(2, 1), \\ C \in V(3, 2) \end{array} \right\}$

Beispiel (3)

Sei $w = b b d d c$ und $G = (\{S, A, B\}, \{b, c, d\}, P, S)$ mit
 $P = \{S \rightarrow AC, A \rightarrow BE, A \rightarrow BD, E \rightarrow AD, C \rightarrow c, B \rightarrow b, D \rightarrow d\}$



3-fach Schleife: $j=3, i=2, k=2$: $V(2, 3) = V(2, 3) \cup \left\{ A \mid \begin{array}{l} A \rightarrow BC, \\ B \in V(2, 2), \\ C \in V(4, 1) \end{array} \right\}$

Beispiel (3)

Sei $w = b b d d c$ und $G = (\{S, A, B\}, \{b, c, d\}, P, S)$ mit
 $P = \{S \rightarrow AC, A \rightarrow BE, A \rightarrow BD, E \rightarrow AD, C \rightarrow c, B \rightarrow b, D \rightarrow d\}$

		b	b	d	d	c
		\xrightarrow{i}				
	$V(i, j)$	1	2	3	4	5
$\downarrow j$	1	B	B	D	D	C
	2		A			
	3		E			
	4					
	5					

3-fach Schleife: $j=3, i=3, k=1: V(3, 3) = V(3, 3) \cup \left\{ A \left| \begin{array}{l} A \rightarrow BC, \\ B \in V(3, 1), \\ C \in V(4, 2) \end{array} \right. \right\}$

Beispiel (3)

Sei $w = b b d d c$ und $G = (\{S, A, B\}, \{b, c, d\}, P, S)$ mit
 $P = \{S \rightarrow AC, A \rightarrow BE, A \rightarrow BD, E \rightarrow AD, C \rightarrow c, B \rightarrow b, D \rightarrow d\}$

		b	b	d	d	c
		\xrightarrow{i}				
$V(i, j)$		1	2	3	4	5
1	B	B	D	D	C	
2		A				
3		E				
4						
5						

j

3-fach Schleife: $j=3, i=3, k=2$: $V(3, 3) = V(3, 3) \cup \left\{ A \left| \begin{array}{l} A \rightarrow BC, \\ B \in V(3, 2), \\ C \in V(5, 1) \end{array} \right. \right\}$

Beispiel (3)

Sei $w = bdddc$ und $G = (\{S, A, B\}, \{b, c, d\}, P, S)$ mit
 $P = \{S \rightarrow AC, A \rightarrow BE, A \rightarrow BD, E \rightarrow AD, C \rightarrow c, B \rightarrow b, D \rightarrow d\}$

		b	b	d	d	c
		$i \rightarrow$				
$V(i, j)$		1	2	3	4	5
$j \downarrow$	1	B	B	D	D	C
	2		A			
	3		E			
	4	A				
	5					

3-fach Schleife: $j=4, i=1, k=1: V(1, 4) = V(1, 4) \cup \left\{ A \left| \begin{array}{l} A \rightarrow BC, \\ B \in V(1, 1), \\ C \in V(2, 3) \end{array} \right. \right\}$

Beispiel (3)

Sei $w = bdddc$ und $G = (\{S, A, B\}, \{b, c, d\}, P, S)$ mit
 $P = \{S \rightarrow AC, A \rightarrow BE, A \rightarrow BD, E \rightarrow AD, C \rightarrow c, B \rightarrow b, D \rightarrow d\}$

		b	b	d	d	c
		\xrightarrow{i}				
	$V(i, j)$	1	2	3	4	5
$\downarrow j$	1	B	B	D	D	C
	2		A			
	3		E			
	4	A				
	5					

3-fach Schleife: $j=4, i=1, k=2$: $V(1, 4) = V(1, 4) \cup \left\{ A \left| \begin{array}{l} A \rightarrow BC, \\ B \in V(1, 2), \\ C \in V(3, 2) \end{array} \right. \right\}$

Beispiel (3)

Sei $w = b b d d c$ und $G = (\{S, A, B\}, \{b, c, d\}, P, S)$ mit
 $P = \{S \rightarrow AC, A \rightarrow BE, A \rightarrow BD, E \rightarrow AD, C \rightarrow c, B \rightarrow b, D \rightarrow d\}$

		b	b	d	d	c
		\xrightarrow{i}				
$V(i, j)$		1	2	3	4	5
1	B	B	D	D	C	
2		A				
3		E				
4	A					
5						

$\downarrow j$

3-fach Schleife: $j=4, i=1, k=3: V(1, 4) = V(1, 4) \cup \left\{ A \left| \begin{array}{l} A \rightarrow BC, \\ B \in V(1, 3), \\ C \in V(4, 1) \end{array} \right. \right\}$

Beispiel (3)

Sei $w = b b d d c$ und $G = (\{S, A, B\}, \{b, c, d\}, P, S)$ mit
 $P = \{S \rightarrow AC, A \rightarrow BE, A \rightarrow BD, E \rightarrow AD, C \rightarrow c, B \rightarrow b, D \rightarrow d\}$

		b	b	d	d	c
		\xrightarrow{i}				
	$V(i, j)$	1	2	3	4	5
j	1	B	B	D	D	C
	2		A			
	3		E			
	4	A				
	5					

3-fach Schleife: $j=4, i=2, k=1$: $V(2, 4) = V(2, 4) \cup \left\{ A \left| \begin{array}{l} A \rightarrow BC, \\ B \in V(2, 1), \\ C \in V(3, 3) \end{array} \right. \right\}$

Beispiel (3)

Sei $w = bdddc$ und $G = (\{S, A, B\}, \{b, c, d\}, P, S)$ mit
 $P = \{S \rightarrow AC, A \rightarrow BE, A \rightarrow BD, E \rightarrow AD, C \rightarrow c, B \rightarrow b, D \rightarrow d\}$

		b	b	d	d	c
		\xrightarrow{i}				
	$V(i, j)$	1	2	3	4	5
$\downarrow j$	1	B	B	D	D	C
	2		A			
	3		E			
	4	A				
	5					

3-fach Schleife: $j=4, i=2, k=2$: $V(2, 4) = V(2, 4) \cup \left\{ A \left| \begin{array}{l} A \rightarrow BC, \\ B \in V(2, 2), \\ C \in V(4, 2) \end{array} \right. \right\}$

Beispiel (3)

Sei $w = bddc$ und $G = (\{S, A, B\}, \{b, c, d\}, P, S)$ mit
 $P = \{S \rightarrow AC, A \rightarrow BE, A \rightarrow BD, E \rightarrow AD, C \rightarrow c, B \rightarrow b, D \rightarrow d\}$

		b	b	d	d	c
		$\xrightarrow{\quad i \quad}$				
	$V(i, j)$	1	2	3	4	5
$\downarrow j$	1	B	B	D	D	C
	2		A			
	3		E			
	4	A				
	5					

3-fach Schleife: $j=4, i=2, k=3$: $V(2, 4) = V(2, 4) \cup \left\{ A \left| \begin{array}{l} A \rightarrow BC, \\ B \in V(2, 3), \\ C \in V(5, 1) \end{array} \right. \right\}$

Beispiel (3)

Sei $w = b b d d c$ und $G = (\{S, A, B\}, \{b, c, d\}, P, S)$ mit
 $P = \{S \rightarrow AC, A \rightarrow BE, A \rightarrow BD, E \rightarrow AD, C \rightarrow c, B \rightarrow b, D \rightarrow d\}$

		b	b	d	d	c
		\xrightarrow{i}				
$V(i, j)$		1	2	3	4	5
1	B	B	D	D	C	
2		A				
3		E				
4	A					
5						

3-fach Schleife: $j=5, i=1, k=1: V(1, 5) = V(1, 5) \cup \left\{ A \left| \begin{array}{l} A \rightarrow BC, \\ B \in V(1, 1), \\ C \in V(2, 4) \end{array} \right. \right\}$

Beispiel (3)

Sei $w = b b d d c$ und $G = (\{S, A, B\}, \{b, c, d\}, P, S)$ mit
 $P = \{S \rightarrow AC, A \rightarrow BE, A \rightarrow BD, E \rightarrow AD, C \rightarrow c, B \rightarrow b, D \rightarrow d\}$

		b	b	d	d	c
		\xrightarrow{i}				
$V(i, j)$		1	2	3	4	5
1	B	B	D	D	C	
2		A				
3		E				
4	A					
5						

j

3-fach Schleife: $j=5, i=1, k=2: V(1, 5) = V(1, 5) \cup \left\{ A \left| \begin{array}{l} A \rightarrow BC, \\ B \in V(1, 2), \\ C \in V(3, 3) \end{array} \right. \right\}$

Beispiel (3)

Sei $w = b b d d c$ und $G = (\{S, A, B\}, \{b, c, d\}, P, S)$ mit
 $P = \{S \rightarrow AC, A \rightarrow BE, A \rightarrow BD, E \rightarrow AD, C \rightarrow c, B \rightarrow b, D \rightarrow d\}$

		b	b	d	d	c
		$i \rightarrow$				
$V(i, j)$		1	2	3	4	5
$j \downarrow$	1	B	B	D	D	C
	2		A			
	3		E			
	4	A				
	5					

3-fach Schleife: $j=5, i=1, k=3: V(1, 5) = V(1, 5) \cup \left\{ A \left| \begin{array}{l} A \rightarrow BC, \\ B \in V(1, 3), \\ C \in V(4, 2) \end{array} \right. \right\}$

Beispiel (3)

Sei $w = b b d d c$ und $G = (\{S, A, B\}, \{b, c, d\}, P, S)$ mit
 $P = \{S \rightarrow AC, A \rightarrow BE, A \rightarrow BD, E \rightarrow AD, C \rightarrow c, B \rightarrow b, D \rightarrow d\}$

		b	b	d	d	c
		\xrightarrow{i}				
	$V(i, j)$	1	2	3	4	5
1	B	B	D	D	C	
2		A				
3		E				
4	A					
5	S					

$\downarrow j$

3-fach Schleife: $j=5, i=1, k=4: V(1, 5) = V(1, 5) \cup \left\{ A \left| \begin{array}{l} A \rightarrow BC, \\ B \in V(1, 4), \\ C \in V(5, 1) \end{array} \right. \right\}$

Beispiel (3)

Sei $w = b b d d c$ und $G = (\{S, A, B\}, \{b, c, d\}, P, S)$ mit
 $P = \{S \rightarrow AC, A \rightarrow BE, A \rightarrow BD, E \rightarrow AD, C \rightarrow c, B \rightarrow b, D \rightarrow d\}$

		b	b	d	d	c
		\xrightarrow{i}				
	$V(i, j)$	1	2	3	4	5
$\downarrow j$	1	B	B	D	D	C
	2		A			
	3		E			
	4	A				
	5	S				

Da $S \in V(1, 5)$ gilt $w \in L(G)$

Kellerautomaten: Motivation

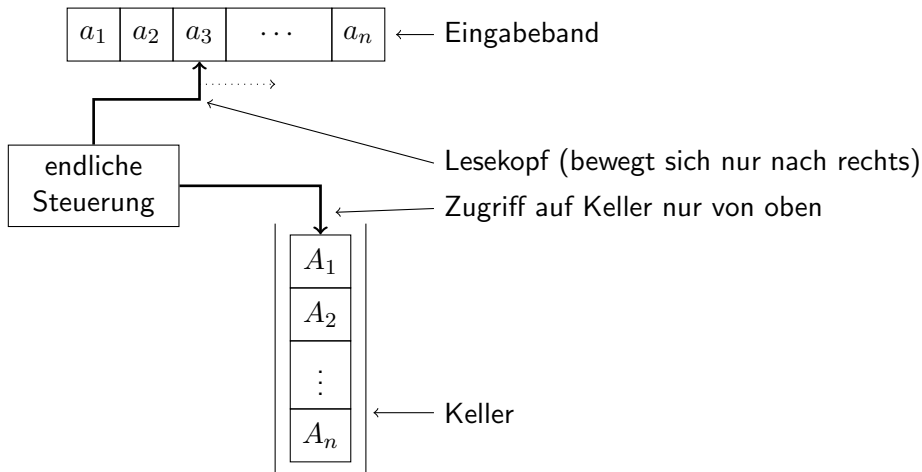
- Endliche Automaten (DFA & NFA) haben fast **keinen Speicher**
- Einziger Speicher dort: Zustände
- Daher endlicher Speicher
- Daher z.B. unmöglich $\{w\bar{w} \mid w \in \{a, b, c\}^*\}$ zu erkennen:
Man müsste beim Lesen von w alle gelesenen Zeichen speichern, um sie dann beim Lesen von \bar{w} zu vergleichen.

Kellerautomaten: Fügen einen Speicher hinzu

- Kellerautomaten haben Kellerspeicher (Stack, LIFO-Speicher, last-in-first-out-Speicher)
- Uendlich großer Speicher als Stapel auf den **nur von oben** zugegriffen werden kann.
- Zustandsübergang:

	Endlicher Automat	Kellerautomat
Eingabe	Zustand und Zeichen	Zustand, Zeichen, oberstes Symbol im Keller
Ausgabe	nächster Zustand	Zustand, Sequenz von Kellersymbolen, die das erste Symbol ersetzen

Kellerautomat: Illustration



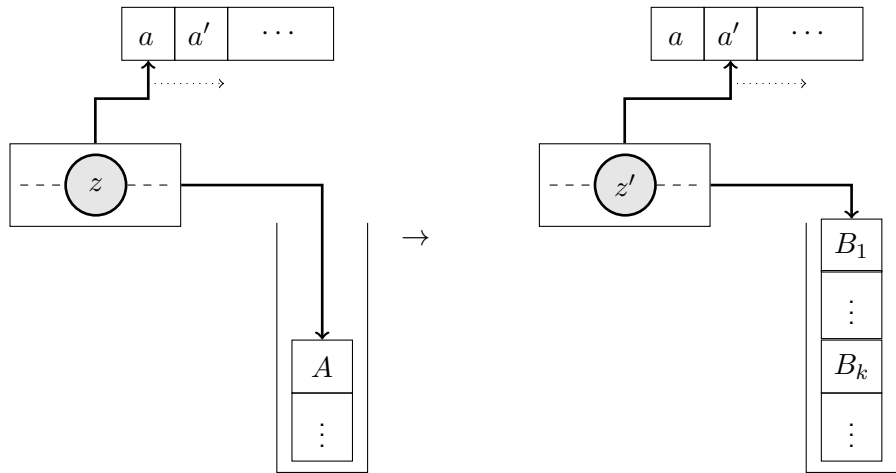
Definition (Kellerautomat, PDA)

Ein (nichtdeterministischer) **Kellerautomat** (PDA, pushdown automaton) ist ein Tupel $M = (Z, \Sigma, \Gamma, \delta, z_0, \#)$, wobei

- Z ist eine endliche Menge von **Zuständen**,
- Σ ist das (endliche) **Eingabealphabet**,
- Γ ist das (endliche) **Kelleralphabet**,
- $\delta : (Z \times (\Sigma \cup \{\varepsilon\}) \times \Gamma) \rightarrow \mathcal{P}_e(Z \times \Gamma^*)$ ist die **Zustandsüberföhrungsfunktion** (oder nur **Überföhrungsfunktion**)
- $z_0 \in Z$ ist der **Startzustand** und
- $\# \in \Gamma$ ist das **Startsymbol im Keller**.

$(z', B_1 \cdots B_k) \in \delta(z, a, A)$ bedeutet: im Zustand z bei Eingabe a und A oben auf dem Keller darf der PDA in Zustand z' wechseln: Dabei wird A durch $B_1 \cdots B_k$ ersetzt (B_1 liegt oben; $k = 0$ ist erlaubt)

Illustration: Zustandsübergang



$$(z', B_1 \dots B_k) \in \delta(z, a, A)$$

Mit unserer Definition von PDAs:

- PDAs sind nichtdeterministisch
- erlauben ε -Übergänge
- keine Endzustände!

Wir werden sehen:

- Akzeptieren: Wenn Eingabe verarbeitet und Keller leer
- Am Anfang: Keller enthält $\#$

- Buchführen während einer Berechnung mit dem PDA:
aktueller Zustand, Resteingabe, aktueller Kellerinhalt
- Wird dargestellt durch PDA-Konfiguration

Definition (Konfiguration eines Kellerautomaten)

Sei $M = (Z, \Sigma, \Gamma, \delta, z_0, \#)$ ein PDA.

Eine Konfiguration von M ist ein Tripel (z, w, W)

mit $z \in Z$, $w \in \Sigma^*$, $W \in \Gamma^*$.

Die Menge aller Konfigurationen für M ist daher $Z \times \Sigma^* \times \Gamma^*$.

- z ist der aktuelle Zustand
- w ist die Resteingabe
- W ist der Kellerinhalt

Definition (Transitionsrelation für PDA-Konfigurationen)

Für einen PDA $M = (Z, \Sigma, \Gamma, \delta, z_0, \#)$ definieren wir

$\vdash_M \subseteq (Z \times \Sigma^* \times \Gamma^*) \times (Z \times \Sigma^* \times \Gamma^*)$ durch

- $(z, a_1 \cdots a_n, A_1 \cdots A_m) \vdash_M (z', a_2 \cdots a_n, W A_2 \cdots A_m)$ falls $(z', W) \in \delta(z, a_1, A_1)$ und
- $(z, w, A_1 \cdots A_m) \vdash_M (z', w, W A_2 \cdots A_m)$ falls $(z', W) \in \delta(z, \varepsilon, A_1)$.

Weitere Notation:

- \vdash_M^* = reflexiv-transitive Hülle von \vdash_M
- \vdash_M^i = i -fache Anwendung von \vdash_M
- Wenn M eindeutig: \vdash statt \vdash_M

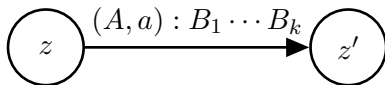
Definition (Akzeptierte Sprache eines PDA)

Sei $M = (Z, \Sigma, \Gamma, \delta, z_0, \#)$ ein PDA. Die durch M **akzeptierte Sprache** $L(M)$ ist definiert als

$$L(M) := \{w \in \Sigma^* \mid (z_0, w, \#) \vdash^* (z, \varepsilon, \varepsilon) \text{ f\"ur ein } z \in Z\}.$$

Notation als Zustandsgraph

- Darstellung analog zu DFA / NFA
- Für $(z', B_1 \cdots B_k) \in \delta(z, a, A)$ zeichnen wir



- Beachte, dass das Startsymbol im Keller bekannt sein muss (überlicherweise $\#$)

Beispiel

PDA $M = (\{z_0, z_1\}, \{a, b\}, \{B, \#\}, \delta, z_0, \#)$ mit

$$\delta(z_0, a, \#) = \{(z_0, B\#)\} \quad \delta(z_0, b, B) = \{(z_1, \varepsilon)\} \quad \delta(z_1, \varepsilon, \#) = \{(z_1, \varepsilon)\}$$

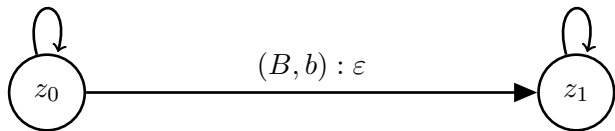
$$\delta(z_0, a, B) = \{(z_0, BB)\} \quad \delta(z_1, b, B) = \{(z_1, \varepsilon)\} \quad \delta(z_0, \varepsilon, \#) = \{(z_0, \varepsilon)\}$$

und $\delta(z_i, c, A) = \emptyset$ in allen anderen Fällen

Zustandsgraph dazu:

$$(\#, a) : B\#, \quad (\#, \varepsilon) : \varepsilon, \quad (B, a) : BB$$

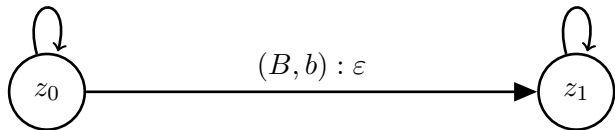
$$(B, b) : \varepsilon, \quad (\#, \varepsilon) : \varepsilon$$



Beispiel (2)

$(\#, a) : B\#$, $(\#, \varepsilon) : \varepsilon$, $(B, a) : BB$

$(B, b) : \varepsilon$, $(\#, \varepsilon) : \varepsilon$



- M akzeptiert ε , denn $(z_0, \varepsilon, \#) \vdash (z_0, \varepsilon, \varepsilon)$.
- M akzeptiert das Wort $a^i b^i$ für $i > 0$, da

$$(z_0, a^i b^i, \#) \vdash (z_0, a^{i-1} b^i, B\#) \vdash^* (z_0, b^i, B^i \#) \\ \vdash (z_1, b^{i-1}, B^{i-1} \#) \vdash^* (z_1, \varepsilon, \#) \vdash (z_1, \varepsilon, \varepsilon).$$

- andere Worte werden nicht akzeptiert:
 - für jedes gelesene a wird ein B auf den Keller gelegt, das durch lesen von b abgebaut werden muss.
 - In z_0 können nur a 's und **ein** b gelesen werden, dann Wechsel in z_1 und dort können nur b 's gelesen werden.
- $L(M) = \{a^i b^i \mid i \in \mathbb{N}\}$

Weiteres Beispiel

Sei $M = (\{z_0, z_1\}, \{a, b\}, \{A, B, \#\}, \delta, z_0, \#)$ mit

$$\begin{array}{ll} \delta(z_0, a, \#) = \{(z_0, A\#), (z_1, \#)\} & \delta(z_0, \varepsilon, A) = \{(z_1, A)\} \\ \delta(z_0, b, \#) = \{(z_0, B\#), (z_1, \#)\} & \delta(z_0, \varepsilon, B) = \{(z_1, B)\} \\ \delta(z_0, a, A) = \{(z_0, AA), (z_1, A)\} & \delta(z_0, \varepsilon, \#) = \{(z_1, \#)\} \\ \delta(z_0, b, A) = \{(z_0, BA), (z_1, A)\} & \delta(z_1, a, A) = \{(z_1, \varepsilon)\} \\ \delta(z_0, a, B) = \{(z_0, AB), (z_1, B)\} & \delta(z_1, b, B) = \{(z_1, \varepsilon)\} \\ \delta(z_0, b, B) = \{(z_0, BB), (z_1, B)\} & \delta(z_1, \varepsilon, \#) = \{(z_1, \varepsilon, \varepsilon)\} \end{array}$$

und $\delta(z_i, c, C) = \emptyset$ für alle anderen Fälle.

$L(M) = \{w \in \{a, b\}^* \mid w \text{ ist Palindrom}\}$:

- In z_0 werden die gelesenen Zeichen (als A, B) auf den Keller gelegt
- In z_1 werden sie dann wieder abgearbeitet (durch Lesen von a, b)
- Wechsel von z_0 zu z_1 mit einem Zeichen (für Palindrome $ua\bar{u}$, $ub\bar{u}$) oder mit ε (für Palindrome $u\bar{u}$).
- Richtiger Zeitpunkt des Wechsels: Macht der Nichtdeterminismus.

Definition (PDA mit Endzuständen)

Ein (nichtdeterministischer) **Kellerautomat mit Endzuständen** (PDA mit Endzuständen) ist ein Tupel $M = (Z, \Sigma, \Gamma, \delta, z_0, \#, E)$ wobei

- Z ist eine endliche Menge von **Zuständen**,
- Σ ist das (endliche) **Eingabealphabet**,
- Γ ist das (endliche) **Kelleralphabet**
- $\delta : Z \times ((\Sigma \cup \{\varepsilon\}) \times \Gamma) \rightarrow \mathcal{P}_e(Z \times \Gamma^*)$ ist die **Überföhrungsfunktion**
- $z_0 \in Z$ ist der **Startzustand**,
- $\# \in \Gamma$ ist das **Startsymbol im Keller** und
- $E \subseteq Z$ ist die Menge der **Endzustände**.

Ein PDA mit Endzuständen akzeptiert die Sprache

$$L(M) = \{w \in \Sigma^* \mid (z_0, w, \#) \vdash^* (z, \varepsilon, W) \text{ und } z \in E\}.$$

Lemma

Für jeden Kellerautomat mit Endzuständen M kann ein Kellerautomat M' (ohne Endzustände) konstruiert werden, so dass $L(M) = L(M')$ gilt

Lemma

Für jeden Kellerautomat M kann ein Kellerautomat mit Endzuständen M' konstruiert werden, so dass $L(M) = L(M')$ gilt.

Satz

PDA's mit Endzuständen und PDA's ohne Endzustände (mit Akzeptanz durch leeren Keller) sind äquivalente Formalismen.

Beweise dazu sind im Skript

Äquivalenz: PDAs und CFLs

- Wir zeigen, dass PDAs genau die Typ 2-Sprachen erkennen.
- Beweis in zwei Teilen:
 - ① Konstruktion eines PDA aus CFG in Greibach-NF
 - ② Konstruktion einer CFG aus einem PDA
(sogenannte **Tripelkonstruktion**)
(PDA mit Einschränkung:
max. 2 Kellersymbole pro Schritt erzeugen)

Ideen:

- CFG in Greibach-Normalform gegeben
- PDA simuliert Linksableitung $S \Rightarrow w$
- Da CFG in Greibach-NF, sieht eine Linksableitung nach i -Schritten immer so aus: $S \Rightarrow^i a_1 \cdots a_i B_1 \cdots B_j$.
- Start mit Eingabe w und S auf dem Keller
- Nach i Schritten, ist $a_1 \cdots a_i$ verarbeitet und $B_1 \cdots B_j$ auf dem Keller

Satz

Jede kontextfreie Sprache wird durch einen Kellerautomaten erkannt.

Beweis:

- Sei L eine CFL und $G = (V, \Sigma, P, S)$ mit $L(G) = L \setminus \{\varepsilon\}$ in Greibach-NF.
- Sei $M = (\{z_0\}, \Sigma, V, \delta, z_0, S)$ ein PDA, sodass

$$\delta(z_0, a, A) := \{(z_0, B_1 \cdots B_n) \mid (A \rightarrow aB_1 \cdots B_n) \in P\}$$

und falls $\varepsilon \in L$ setze zusätzlich $\delta(z_0, \varepsilon, S) := \{(z_0, \varepsilon)\}$.

In allen anderen Fällen sei $\delta(z_0, \varepsilon, A) = \emptyset$.

- Wir zeigen $L(M) = L$.
- Zunächst: $\varepsilon \in L$ g.d.w. $(z_0, \varepsilon, S) \vdash (z_0, \varepsilon, \varepsilon)$ und damit $\varepsilon \in L(M)$.

CFG \rightarrow PDA (2)

$M = (\{z_0\}, \Sigma, \Sigma \cup V, \delta, z_0, S)$ mit $\delta(z_0, a, A) := \{(z_0, B_1 \cdots B_n) \mid (A \rightarrow aB_1 \cdots B_n) \in P\} \dots$

Beweis (Fortsetzung):

- Für die weiteren Fälle zeigen wir für alle $i \in \mathbb{N}$ (mit Induktion über i)

$S \Rightarrow_G^i a_1 \cdots a_i B_1 \cdots B_m$ mit einer Linksableitung

genau dann, wenn

$(z_0, a_1 \cdots a_i w, S) \vdash^i (z_0, w, B_1 \cdots B_m)$ für alle $w \in \Sigma^*$.

- Basis $i = 0$: gilt, denn $S \Rightarrow_G^0 S$ und $(z_0, w, S) \vdash^0 (z_0, w, S)$

- Für $i > 0$ und “ \Rightarrow ”:

- Sei $S \Rightarrow_G^i a_1 \cdots a_i B_1 \cdots B_m$ eine Linksableitung.

- Da G in Greibach-NF, kann diese geschrieben werden als

$S \Rightarrow_G^{i-1} a_1 \cdots a_{i-1} B_x B_{j+1} \cdots B_m \Rightarrow_G a_1 \cdots a_i B_1 \cdots B_m$, wobei $B_x \rightarrow a_i B_1 \cdots B_j \in P$ als letzte Produktion angewendet wurde.

- Induktionsannahme liefert: $S \Rightarrow_G^{i-1} a_1 \cdots a_{i-1} B_x B_{j+1} \cdots B_m$ genau dann, wenn $(z_0, a_1 \cdots a_{i-1} w, S) \vdash^{i-1} (z_0, w, B_x B_{j+1} \cdots B_k)$.

- Verwende $w = a_i w'$ und da $(z_0, B_1 \cdots B_j) \in \delta(z_0, a_i, B_x)$, gilt $(z_0, a_1 \cdots a_i w', S) \vdash^i (z_0, w', B_1 \cdots B_k)$ für alle w' .

CFG \rightarrow PDA (3)

$M = (\{z_0\}, \Sigma, \Sigma \cup V, \delta, z_0, S)$ mit $\delta(z_0, a, A) := \{(z_0, B_1 \cdots B_n) \mid (A \rightarrow aB_1 \cdots B_n) \in P\} \dots$

Beweis (Fortsetzung):

- Für $i > 0$ und " \Leftarrow ":
 - Sei $(z_0, a_1 \cdots a_i w, S) \vdash^i (z_0, w, B_1 \cdots B_k)$.
 - Dann muss der letzte Schritt a_i gelesen haben
 - D.h. die Folge lässt sich zerlegen in

$$(z_0, a_1 \cdots a_i w, S) \vdash^{i-1} (z_0, a_i w, B_x B_{j+1} \cdots B_k) \vdash (z_0, w, B_1 \cdots B_k),$$

wobei $(z_0, B_1 \cdots B_j) \in \delta(z_0, a_i, B_x)$.

- Dann muss $B_x \rightarrow aB_1 \cdots B_j$ eine Produktion in P sein.
- Induktionsannahme liefert: $S \Rightarrow_G^{i-1} a_1 \cdots a_{i-1} B_x B_{j+1} \cdots B_k$ und wir können obige Produktion anwenden und erhalten $S \Rightarrow_G^i a_1 \cdots a_i B_1 \cdots B_k$. □

Lemma (PDAs mit Erzeugung von ≤ 2 Kellersymbolen)

Für jeden PDA $M = (Z, \Sigma, \Gamma, \delta, z_0, \#)$ gibt es einen PDA $M' = (Z, \Sigma, \Gamma', \delta', z_0, \#)$ mit $L(M) = L(M')$, sodass gilt: Wenn $(z', B_1 \cdots B_k) \in \delta'(z, a, A)$ (für $a \in (\Sigma \cup \{\varepsilon\})$), dann ist $k \leq 2$.

Beweis (Skizze):

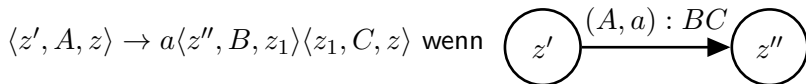
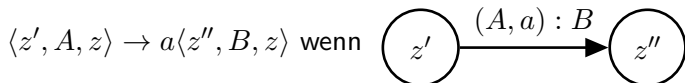
Transformiere M in M' wie folgt (mit $A \in \Gamma$ und $a \in (\Sigma \cup \{\varepsilon\})$):

- $(z', B_1 \cdots B_k) \in \delta'(z, a, A)$ wenn $(z', B_1 \cdots B_k) \in \delta(z, a, A)$, $k \leq 2$.
- falls $(z', B_1 \cdots B_k) \in \delta(z, a, A)$ mit $k > 2$, dann
 - $(z, C_k B_k) \in \delta'(z, a, A)$, und
 - $\delta(z, \varepsilon, C_i) = \{(z, C_{i-1} B_{i-1})\}$ für alle i mit $4 \leq i \leq k$, und
 - $\delta(z, \varepsilon, C_3) = \{(z', B_1 B_2)\}$

wobei $C_3, \dots, C_k \in \Gamma'$ neue Kellersymbole sind
(diese werden jeweils neu erzeugt pro ersetztem Eintrag).

Ideen

- Verwende PDA mit Erzeugung von ≤ 2 Kellersymbolen
- Erzeuge Grammatik mit Tripelkonstruktion
- Variablen der Grammatik:
Tripel $\langle z', A, z \rangle$, die alle Worte w erzeugt, die den PDA von z' mit Kellerinhalt A und Wort w zu z und leeren Keller führen
- Produktionen



Satz

Kellerautomaten akzeptieren kontextfreie Sprachen.

Beweis: Sei $M = (Z, \Sigma, \Gamma, \delta, z_0, \#)$ ein PDA mit $k \leq 2$ für alle $(z', B_1 \cdots B_k) \in \delta(z, a, A)$ (und $a \in (\Sigma \cup \{\varepsilon\})$).

Konstruiere $G = (V, \Sigma, P, S)$ mit S neues Symbol und

$$V = \{S\} \cup \{\langle z_i, A, z_j \rangle \mid z_i, z_j \in Z, A \in \Gamma\}$$

$$P = \{S \rightarrow \langle z_0, \#, z \rangle \mid z \in Z\}$$

$$\cup \{\langle z', A, z \rangle \rightarrow a \mid (z, \varepsilon) \in \delta(z', a, A), a \in \Sigma \cup \{\varepsilon\}, A \in \Gamma\}$$

$$\cup \{\langle z', A, z \rangle \rightarrow a \langle z'', B, z \rangle \mid (z'', B) \in \delta(z', a, A), z \in Z, a \in \Sigma \cup \{\varepsilon\}, A \in \Gamma\}$$

$$\cup \{\langle z', A, z \rangle \rightarrow a \langle z'', B, z_1 \rangle \langle z_1, C, z \rangle \mid (z'', BC) \in \delta(z', a, A), \\ z, z_1 \in Z, a \in \Sigma \cup \{\varepsilon\}, A \in \Gamma\}$$

Wir beweisen $\langle z', A, z \rangle \Rightarrow_G^* w$ g.d.w. $(z', w, A) \vdash_M^* (z, \varepsilon, \varepsilon)$.

Da $S \rightarrow \langle z_0, A, z \rangle$ folgt: $w \in L(G) \iff w \in L(M)$, d. h. $L(G) = L(M)$.

„ \Rightarrow “:

- Sei $\langle z', A, z \rangle \Rightarrow_G^i w$ eine Linksableitung.
- Wir verwenden Induktion über i .
- Basis $i = 1$: Sei $\langle z', A, z \rangle \Rightarrow_G w$
 - Verwendete Produktion muss $\langle z', A, z \rangle \rightarrow a$ sein
 - Dann muss $(z, \varepsilon) \in \delta(z', a, A)$ gelten und damit gilt: $(z', a, A) \vdash (z, \varepsilon, \varepsilon)$.
- Schritt: $\langle z', A, z \rangle \Rightarrow_G u \Rightarrow_G^{i-1} w$ mit $i - 1 > 0$
 - Wenn $u = a \in (\Sigma \cup \{\varepsilon\})$, dann kann $i - 1 > 0$ nicht gelten.
 - Wenn $u = a \langle z'', B, z \rangle$, dann $(z'', B) \in \delta(z', a, A)$ und $u = a \langle z'', B, z \rangle \Rightarrow^{i-1} aw' = w$.
Dann gilt $\langle z'', B, z \rangle \Rightarrow^{i-1} w'$ und die Induktionsannahme liefert $(z'', w', B) \vdash_M^* (z, \varepsilon, \varepsilon)$. Mit $(z'', B) \in \delta(z', a, A)$ zeigt dies $(z', u, A) = (z', aw', A) \vdash_M (z'', w', B) \vdash_M^* (z, \varepsilon, \varepsilon)$.

...

- Wenn $u = a\langle z'', B, z_1 \rangle \langle z_1, C, z \rangle$, dann ist $(z'', BC) \in \delta(z', a, A)$ und $u = a\langle z'', B, z_1 \rangle \langle z_1, C, z \rangle \Rightarrow^{i-1} aw' = w$

Dann gilt auch $\langle z'', B, z_1 \rangle \langle z_1, C, z \rangle \Rightarrow^{i-1} w'$ und es gibt Linksableitungen $\langle z'', B, z_1 \rangle \Rightarrow^j w'_0$ und $\langle z_1, C, z \rangle \Rightarrow^k w'_1$ mit $j + k \leq i - 1$, $w' = w'_0 w'_1$.

Für beide können wir die Induktionsannahme anwenden und erhalten $(z'', w'_0, B) \vdash_M^* (z_1, \varepsilon, \varepsilon)$ und $(z_1, w'_1, C) \vdash_M^* (z, \varepsilon, \varepsilon)$.

Abändern der 1. Konfigurationsfolge: C auf den Keller & w'_1 anhängen
 $(z'', w', BC) = (z'', w'_0 w'_1, BC) \vdash_M^* (z_1, w'_1, C)$.

Anhängen der 2. Konfigurationsfolge liefert: $(z'', w', BC) \vdash_M^* (z, \varepsilon, \varepsilon)$.

Da $(z'', BC) \in \delta(z', a, A)$, gilt

$$(z', u, BC) = (z', aw', BC) \vdash_M (z'', w', BC) \vdash_M^* (z, \varepsilon, \varepsilon).$$

„ \Leftarrow “:

- Sei $\langle z', w, A \rangle \vdash_M^i (z, \varepsilon, \varepsilon)$. Zeige $\langle z', A, z \rangle \Rightarrow_G^* w$ mit Induktion über i
- Basis $i = 1$: Dann gilt $w = a \in (\Sigma \cup \{\varepsilon\})$ und $(z, \varepsilon) \in \delta(z', w, A)$.
Damit gibt es $\langle z', A, z \rangle \rightarrow a \in P$ und daher $\langle z', A, z \rangle \Rightarrow_G a$.
- Schritt: Sei $i > 1$ und daher $(z', aw', A) \vdash (z'', w', \alpha) \vdash_M^{i-1} (z, \varepsilon, \varepsilon)$ für $i - 1 > 0$, $a \in \Sigma \cup \{\varepsilon\}$ und $\alpha = \varepsilon$, $\alpha = B$ oder $\alpha = BC$.

Wir betrachten alle drei Fälle für α einzeln:

- $\alpha = \varepsilon$: Dieser Fall ist nicht möglich, da $i - 1 > 0$ nicht gelten kann.
- $\alpha = B$. Dann ist $\langle z', A, z \rangle \rightarrow a \langle z'', B, z \rangle \in P$.

Da $(z'', w', B) \vdash_M^{i-1} (z, \varepsilon, \varepsilon)$, liefert Induktionsannahme $\langle z'', B, z \rangle \Rightarrow_G^* w'$ und daher: $\langle z', A, z \rangle \Rightarrow_G a \langle z'', B, z \rangle \Rightarrow_G^* aw' = w$.

...

- $\alpha = BC$. Dann ist $\langle z', A, z \rangle \rightarrow \langle z'', B, z_1 \rangle \langle z_1, C, z \rangle \in P$.

Schreibe $(z'', w', BC) \vdash_M^{i-1} (z, \varepsilon, \varepsilon)$ als

$(z'', w'_1 w'_2, BC) \vdash_M^j (z_1, w'_2, C) \vdash_M^k (z, \varepsilon, \varepsilon)$ mit $j + k = i - 1$.

Weglassen von C und w'_2 im ersten Teil zeigt:

$(z'', w'_1, B) \vdash_M^j (z_1, \varepsilon, \varepsilon)$,

Da $j < i$ und $k < i$ liefert Induktionsannahme

$\langle z'', B, z_1 \rangle \Rightarrow_G^* w'_1$ und $\langle z_1, C, z \rangle \Rightarrow_G^* w'_2$.

Daher gilt

$\langle z', A, z \rangle \Rightarrow_G a \langle z'', B, z_1 \rangle \langle z_1, C, z \rangle \Rightarrow_G^* a w'_1 \langle z_1, C, z \rangle \Rightarrow_G^* a w'_1 w'_2 = w$.

Die gezeigten Sätze zusammengefasst ergeben:

Theorem

Kellerautomaten erkennen genau die kontextfreien Sprachen.

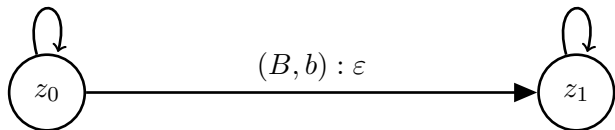
Die bisherigen Beweise zeigen auch, dass man PDAs einschränken kann auf PDAs mit genau einem Zustand:

- Sei M ein PDA.
- Transformiere M in Grammatik G mit $L(G) = L(M)$
- Transformiere G in G' in Greibach-Normalform (mit $L(G') = L(G) \setminus \varepsilon$)
- Transformiere Grammatik G' in PDA M' mit $L(M') = L(G)$
unsere Konstruktion verwendet nur einen Zustand!

Beispiel

$(\#, a) : B\#, (\#, \varepsilon) : \varepsilon, (B, a) : BB$

$(B, b) : \varepsilon, (\#, \varepsilon) : \varepsilon$



Der vorherige Beweis konstruiert die Grammatik $G = (V, \Sigma, P, S)$ mit

$V = \{S, \langle z_0, B, z_0 \rangle, \langle z_0, B, z_1 \rangle, \langle z_1, B, z_0 \rangle, \langle z_1, B, z_1 \rangle,$
 $\langle z_0, \#, z_0 \rangle, \langle z_0, \#, z_1 \rangle, \langle z_1, \#, z_0 \rangle, \langle z_1, \#, z_1 \rangle\}$

$P = \{S \rightarrow \langle z_0, \#, z_0 \rangle, S \rightarrow \langle z_0, \#, z_1 \rangle.\}$

$\cup \{\langle z_0, B, z_1 \rangle \rightarrow b, \langle z_1, B, z_1 \rangle \rightarrow b, \langle z_0, \#, z_0 \rangle \rightarrow \varepsilon, \langle z_1, \#, z_1 \rangle \rightarrow \varepsilon\}$

$\cup \{\langle z_0, \#, z_0 \rangle \rightarrow a\langle z_0, B, z_0 \rangle, \langle z_0, \#, z_1 \rangle \rightarrow a\langle z_0, B, z_1 \rangle\}$

$\cup \{\langle z_0, B, z_0 \rangle \rightarrow a\langle z_0, B, z_0 \rangle\langle z_0, B, z_0 \rangle, \langle z_0, B, z_1 \rangle \rightarrow a\langle z_0, B, z_0 \rangle\langle z_0, B, z_1 \rangle,$
 $\langle z_0, B, z_0 \rangle \rightarrow a\langle z_0, B, z_1 \rangle\langle z_1, B, z_0 \rangle, \langle z_0, B, z_1 \rangle \rightarrow a\langle z_0, B, z_1 \rangle\langle z_1, B, z_1 \rangle\}$

Beispiel (2)

Vereinfachen der Grammatik ergibt:

$$\{S \rightarrow \langle z_0, \#, z_0 \rangle \mid \langle z_0, \#, z_1 \rangle, \langle z_0, \#, z_0 \rangle \rightarrow \varepsilon, \langle z_0, \#, z_1 \rangle \rightarrow a\langle z_0, B, z_1 \rangle, \\ \langle z_0, B, z_1 \rangle \rightarrow b \mid a\langle z_0, B, z_1 \rangle\langle z_1, B, z_1 \rangle, \langle z_1, B, z_1 \rangle \rightarrow b\}$$

Umbenennen, Streichen von nicht erreichbaren Variablen und Entfernen von Einheitsproduktionen ergibt

$$G = (\{S, B, C\}, \{a, b\}, \{S \rightarrow \varepsilon \mid aB, B \rightarrow b \mid aBC, C \rightarrow b\}, S)$$

(ist bis auf ε -Produktion in Greibach-NF.)

Beispiel (3)

Der vorherige Beweis konstruiert für

$$G = (\{S, B, C\}, \{a, b\}, \{S \rightarrow \varepsilon \mid aB, B \rightarrow b \mid aBC, C \rightarrow b\}, S)$$

den PDA $M = (\{z_0\}, \Sigma, \Sigma \cup V, \delta, z_0, S)$ mit

$$\begin{aligned} \delta(z_0, a, S) &= \{(z_0, B)\} & \delta(z_0, b, B) &= \{(z_0, \varepsilon)\} & \delta(z_0, a, B) &= \{(z_0, BC)\} \\ \delta(z_0, b, C) &= \{(z_0, \varepsilon)\} & \delta(z_0, \varepsilon, S) &= \{(z_0, \varepsilon)\} & \delta(z_0, d, A) &= \emptyset \text{ sonst} \end{aligned}$$

Eine Konfigurationsfolge für die Eingabe $aaabbb$ ist

$$(z_0, aaabbb, S)$$

Beispiel (3)

Der vorherige Beweis konstruiert für

$$G = (\{S, B, C\}, \{a, b\}, \{S \rightarrow \varepsilon \mid aB, B \rightarrow b \mid aBC, C \rightarrow b\}, S)$$

den PDA $M = (\{z_0\}, \Sigma, \Sigma \cup V, \delta, z_0, S)$ mit

$$\begin{aligned} \delta(z_0, a, S) &= \{(z_0, B)\} & \delta(z_0, b, B) &= \{(z_0, \varepsilon)\} & \delta(z_0, a, B) &= \{(z_0, BC)\} \\ \delta(z_0, b, C) &= \{(z_0, \varepsilon)\} & \delta(z_0, \varepsilon, S) &= \{(z_0, \varepsilon)\} & \delta(z_0, d, A) &= \emptyset \text{ sonst} \end{aligned}$$

Eine Konfigurationsfolge für die Eingabe $aaabbb$ ist

$$\begin{aligned} &(z_0, aaabbb, S) \\ &\vdash (z_0, aabbb, B) \end{aligned}$$

Beispiel (3)

Der vorherige Beweis konstruiert für

$$G = (\{S, B, C\}, \{a, b\}, \{S \rightarrow \varepsilon \mid aB, B \rightarrow b \mid aBC, C \rightarrow b\}, S)$$

den PDA $M = (\{z_0\}, \Sigma, \Sigma \cup V, \delta, z_0, S)$ mit

$$\begin{aligned} \delta(z_0, a, S) &= \{(z_0, B)\} & \delta(z_0, b, B) &= \{(z_0, \varepsilon)\} & \delta(z_0, a, B) &= \{(z_0, BC)\} \\ \delta(z_0, b, C) &= \{(z_0, \varepsilon)\} & \delta(z_0, \varepsilon, S) &= \{(z_0, \varepsilon)\} & \delta(z_0, d, A) &= \emptyset \text{ sonst} \end{aligned}$$

Eine Konfigurationsfolge für die Eingabe $aaabbb$ ist

$$\begin{aligned} &(z_0, aaabbb, S) \\ &\vdash (z_0, aabbb, B) \\ &\vdash (z_0, abbb, BC) \end{aligned}$$

Beispiel (3)

Der vorherige Beweis konstruiert für

$$G = (\{S, B, C\}, \{a, b\}, \{S \rightarrow \varepsilon \mid aB, B \rightarrow b \mid aBC, C \rightarrow b\}, S)$$

den PDA $M = (\{z_0\}, \Sigma, \Sigma \cup V, \delta, z_0, S)$ mit

$$\begin{aligned} \delta(z_0, a, S) &= \{(z_0, B)\} & \delta(z_0, b, B) &= \{(z_0, \varepsilon)\} & \delta(z_0, a, B) &= \{(z_0, BC)\} \\ \delta(z_0, b, C) &= \{(z_0, \varepsilon)\} & \delta(z_0, \varepsilon, S) &= \{(z_0, \varepsilon)\} & \delta(z_0, d, A) &= \emptyset \text{ sonst} \end{aligned}$$

Eine Konfigurationsfolge für die Eingabe $aaabbb$ ist

$$\begin{aligned} &(z_0, aaabbb, S) \\ &\vdash (z_0, aabbb, B) \\ &\vdash (z_0, abbb, BC) \\ &\vdash (z_0, bbb, BCC) \end{aligned}$$

Beispiel (3)

Der vorherige Beweis konstruiert für

$$G = (\{S, B, C\}, \{a, b\}, \{S \rightarrow \varepsilon \mid aB, B \rightarrow b \mid aBC, C \rightarrow b\}, S)$$

den PDA $M = (\{z_0\}, \Sigma, \Sigma \cup V, \delta, z_0, S)$ mit

$$\begin{aligned} \delta(z_0, a, S) &= \{(z_0, B)\} & \delta(z_0, b, B) &= \{(z_0, \varepsilon)\} & \delta(z_0, a, B) &= \{(z_0, BC)\} \\ \delta(z_0, b, C) &= \{(z_0, \varepsilon)\} & \delta(z_0, \varepsilon, S) &= \{(z_0, \varepsilon)\} & \delta(z_0, d, A) &= \emptyset \text{ sonst} \end{aligned}$$

Eine Konfigurationsfolge für die Eingabe $aaabbb$ ist

$$\begin{aligned} &(z_0, aaabbb, S) \\ &\vdash (z_0, aabbb, B) \\ &\vdash (z_0, abbb, BC) \\ &\vdash (z_0, bbb, BCC) \\ &\vdash (z_0, bb, CC) \end{aligned}$$

Beispiel (3)

Der vorherige Beweis konstruiert für

$$G = (\{S, B, C\}, \{a, b\}, \{S \rightarrow \varepsilon \mid aB, B \rightarrow b \mid aBC, C \rightarrow b\}, S)$$

den PDA $M = (\{z_0\}, \Sigma, \Sigma \cup V, \delta, z_0, S)$ mit

$$\begin{aligned} \delta(z_0, a, S) &= \{(z_0, B)\} & \delta(z_0, b, B) &= \{(z_0, \varepsilon)\} & \delta(z_0, a, B) &= \{(z_0, BC)\} \\ \delta(z_0, b, C) &= \{(z_0, \varepsilon)\} & \delta(z_0, \varepsilon, S) &= \{(z_0, \varepsilon)\} & \delta(z_0, d, A) &= \emptyset \text{ sonst} \end{aligned}$$

Eine Konfigurationsfolge für die Eingabe $aaabbb$ ist

$$\begin{aligned} &(z_0, aaabbb, S) \\ &\vdash (z_0, aabbb, B) \\ &\vdash (z_0, abbb, BC) \\ &\vdash (z_0, bbb, BCC) \\ &\vdash (z_0, bb, CC) \\ &\vdash (z_0, b, C) \end{aligned}$$

Beispiel (3)

Der vorherige Beweis konstruiert für

$$G = (\{S, B, C\}, \{a, b\}, \{S \rightarrow \varepsilon \mid aB, B \rightarrow b \mid aBC, C \rightarrow b\}, S)$$

den PDA $M = (\{z_0\}, \Sigma, \Sigma \cup V, \delta, z_0, S)$ mit

$$\begin{aligned} \delta(z_0, a, S) &= \{(z_0, B)\} & \delta(z_0, b, B) &= \{(z_0, \varepsilon)\} & \delta(z_0, a, B) &= \{(z_0, BC)\} \\ \delta(z_0, b, C) &= \{(z_0, \varepsilon)\} & \delta(z_0, \varepsilon, S) &= \{(z_0, \varepsilon)\} & \delta(z_0, d, A) &= \emptyset \text{ sonst} \end{aligned}$$

Eine Konfigurationsfolge für die Eingabe $aaabbb$ ist

$$\begin{aligned} &(z_0, aaabbb, S) \\ &\vdash (z_0, aabbb, B) \\ &\vdash (z_0, abbb, BC) \\ &\vdash (z_0, bbb, BCC) \\ &\vdash (z_0, bb, CC) \\ &\vdash (z_0, b, C) \\ &\vdash (z_0, \varepsilon, \varepsilon) \end{aligned}$$

Deterministisch kontextfreie Sprachen

- Definiert durch deterministische Kellerautomaten mit **Akzeptanz durch Endzustände**.
- ε -Übergänge sind erlaubt, aber nur wenn es keinen anderen Übergang (mit einem Terminalzeichen und selben Kellersymbol) gibt.

Definition (Deterministischer Kellerautomat, DPDA)

Ein Kellerautomat mit Endzuständen $M = (Z, \Sigma, \Gamma, \delta, z_0, \#, E)$ ist **deterministisch** (ein DPDA) wenn für alle $(z, a, A) \in (Z, \Sigma, \Gamma)$ gilt:

$$|\delta(z, a, A)| + |\delta(z, \varepsilon, A)| \leq 1.$$

Die von DPDAs akzeptierten Sprachen heißen **deterministisch kontextfrei**.

Beispiele (1)

Satz

Die Sprache $L = \{w\$ \bar{w} \mid w \in \{a, b\}^*\}$ ist deterministisch kontextfrei.

Beweis: Betrachte den DPDA

$M = (\{z_0, z_1, z_2\}, \{a, b, \$\}, \{\#, A, B\}, \delta, z_0, \#, \{z_2\})$ mit

$$\begin{array}{ll} \delta(z_0, a, \#) = \{(z_0, A\#)\} & \delta(z_0, \$, A) = \{(z_1, A)\} \\ \delta(z_0, b, \#) = \{(z_0, B\#)\} & \delta(z_0, \$, B) = \{(z_1, B)\} \\ \delta(z_0, a, A) = \{(z_0, AA)\} & \delta(z_0, \$, \#) = \{(z_1, \#)\} \\ \delta(z_0, b, A) = \{(z_0, BA)\} & \delta(z_1, a, A) = \{(z_1, \varepsilon)\} \\ \delta(z_0, a, B) = \{(z_0, AB)\} & \delta(z_1, b, B) = \{(z_1, \varepsilon)\} \\ \delta(z_0, b, B) = \{(z_0, BB)\} & \delta(z_1, \varepsilon, \#) = \{(z_2, \varepsilon)\} \end{array}$$

und $\delta(z_i, c, C) = \emptyset$ sonst

Beachte: $L = \{w\bar{w} \mid w \in \{a, b\}^*\}$ ist **nicht** deterministisch kontextfrei
aber kontextfrei

Beispiele (2)

Satz

Die Sprache $L = \{a^i b^i \mid i \in \mathbb{N}_{>0}\}$ ist deterministisch kontextfrei.

Beweis: Betrachte den DPDA

$M = (\{z_0, z_1, z_2\}, \{a, b\}, \{\#, A\}, \delta, z_0, \#, \{z_2\})$ mit

$$\delta(z_0, a, \#) = \{(z_0, A\#)\}$$

$$\delta(z_0, a, A) = \{(z_0, AA)\}$$

$$\delta(z_0, b, A) = \{(z_1, \varepsilon)\}$$

$$\delta(z_1, b, A) = \{(z_1, \varepsilon)\}$$

$$\delta(z_1, \varepsilon, \#) = \{(z_2, \varepsilon)\}$$

und $\delta(z_i, c, B) = \emptyset$, sonst

Theorem (Eigenschaften determinist. kontextfreier Sprachen)

- 1 Das Wortproblem für deterministisch kontextfreie Sprachen kann in Linearzeit entschieden werden.
- 2 Für deterministisch kontextfreie Sprachen gibt es eindeutige Grammatiken.
- 3 Deterministisch kontextfreie Sprachen sind unter Komplementbildung abgeschlossen.

Beweis: siehe Literatur

Satz

Deterministisch kontextfreie Sprachen sind **nicht** abgeschlossen bezüglich Vereinigung und Schnitt.

Beweis: i) Schnittbildung:

- Die Sprachen $L_1 = \{a^n b^n c^m \mid n, m \in \mathbb{N}_{>0}\}$ und $L_2 = \{a^n b^m c^m \mid n, m \in \mathbb{N}_{>0}\}$ sind deterministisch kontextfrei
- $L_1 \cap L_2 = \{a^n b^n c^n \mid n \in \mathbb{N}_{>0}\}$ ist nicht kontextfrei.

ii) Vereinigung:

- $L \cap L' = \overline{\overline{L} \cup \overline{L'}}$
- Annahme: Det. CFLs abgeschlossen bez. Vereinigung
- Da Det. CFLs auch abgeschlossen bez. Komplement, folgt: Det. CFLs abgeschlossen bez. Schnitt. Widerspruch!
- D.h. Annahme falsch, Det. CFLs nicht abgeschlossen bez. \cup .

Weitere Eigenschaften (2)

Satz

Der Schnitt einer (deterministisch) kontextfreien Sprachen mit einer regulären Sprache ist (deterministisch) kontextfrei.

Beweis: Sei $M = (Z, \Sigma, \Gamma, \delta, z_0, \#, E)$ ein PDA mit Endzuständen und $M' = (Z', \Sigma, \delta', z'_0, E')$ ein DFA. Konstruiere PDA mit Endzuständen: $M'' = (Z \times Z', \Sigma, \delta'', (z_0, z'_0), \#, E \times E')$ mit

- $(z_k, z'_k, B_1 \cdots B_m) \in \delta''((z_i, z'_i), a, A)$ falls $(z_k, B_1 \cdots B_m) \in \delta(z_i, a, A)$ und $\delta'(z'_i, a) = z'_k$ und
- $(z_k, z'_i, B_1 \cdots B_m) \in \delta''((z_i, z'_i), \varepsilon, A)$ falls $(z_k, B_1 \cdots B_m) \in \delta(z_i, \varepsilon, A)$.

Es gilt:

- $L(M'') = L(M) \cap L(M')$, denn M'' simuliert M und M' gleichzeitig, und akzeptiert nur, wenn beide Automaten akzeptieren.
- M'' ist deterministisch, wenn M deterministisch ist.

- CYK-Algorithmus zeigt: Das Wortproblem für kontextfreie Grammatiken ist effizient entscheidbar.
- Viele Fragestellungen sind für CFLs unentscheidbar (z.B. das Äquivalenzproblem und das Schnittproblem)
- Wir betrachten weitere Entscheidungsprobleme

Satz

Das Leerheitsproblem für kontextfreie Grammatiken ist entscheidbar.

Beweis:

- Sei L als CFG gegeben
- Prüfe zunächst, ob $\varepsilon \in L$ (wenn ja, dann ist L nicht leer)
- Sei $G = (V, \Sigma, P, S)$ eine CFG in Chomsky-NF mit $L(G) = L \setminus \{\varepsilon\}$.
- Der folgende Algorithmus markiert alle $A \in V$ mit $\{w \in \Sigma^* \mid A \Rightarrow_G^* w\} \neq \emptyset$
- Prüfe, ob S markiert wird (wenn ja, dann ist L nicht-leer)

Algorithmus 9: Markierung der Variablen, die nichtleere Sprachen erzeugen

Eingabe: Grammatik $G = (V, \Sigma, P, S)$ in Chomsky-Normalform

Ausgabe: Menge $W \subseteq V$ aller Variablen, die nicht die leere Sprache erzeugen

Beginn

$W := \{A \in V \mid A \rightarrow a \in P, a \in \Sigma\};$

wiederhole

$W_{alt} := W;$

$W := W_{alt} \cup \{A \mid A \rightarrow BC \in P, B \in W_{alt}, C \in W_{alt}\};$

bis $W = W_{alt};$

return W

Satz

Das Endlichkeitsproblem für kontextfreie Sprachen ist entscheidbar.

Beweis: Sei $G = (V, \Sigma, P, S)$ eine CFG in Chomsky-NF. Sei n die Zahl aus dem Pumping-Lemma für CFGs (z.B. $n = 2^{|V|}$ siehe Beweis Pumping-L.).

Wir zeigen zunächst: Es gilt $|L(G)| = \infty$ g.d.w. es ein Wort $z \in L(G)$ mit $n \leq |z| < 2n$ gibt:

„ \Leftarrow “:

- Sei $z \in L$ mit $|z| \geq n$.
- Pumping-Lemma zeigt: $uv^iwx^iy \in L$ für alle $i \in \mathbb{N}$.
- Also $|L(G)| = \infty$

Endlichkeitsproblem (2)

...

Wir zeigen zunächst: Es gilt $|L(G)| = \infty$ g.d.w. es ein Wort $z \in L(G)$ mit $n \leq |z| < 2n$ gibt:

„ \Rightarrow “:

- Beweis durch Widerspruch
- Annahme: Es gibt kein Wort $z \in L(G)$ für $n \leq |z| < 2n$, aber trotzdem gilt $|L(G)| = \infty$.
- Sei $z \in L(G)$ das kürzeste Wort mit $|z| \geq 2n$.
- Pumping-Lemma: Es gibt u, v, w, x, y gibt mit $z = uvwxy$, $|vx| > 0$ und $|vwx| \leq n$, sodass insbes. $uv^0wx^0y \in L$ gilt.
- Da $|uv^0wx^0y| = |uwy| < |uvwxy|$ und $|uwy| \geq n$ gilt, war z nicht minimal gewählt. Widerspruch!

Entscheide Endlichkeitsproblem: Teste für alle Worte $w \in \Sigma^*$, der Länge $n \leq |w| < 2n$, ob $w \in L(G)$ gilt (mit CYK-Algorithmus).

Weiteres Entscheidbarkeitsproblem

Das Problem, ob eine deterministisch kontextfreie Sprache äquivalent zu einer regulären Sprache ist, ist entscheidbar.

- Sei L_1 durch DPDA gegeben und L_2 durch einen DFA.
- Prüfe $\overline{L_1} \cap L_2 = \emptyset$ und $L_1 \cap \overline{L_2} = \emptyset$
- Beides ist entscheidbar, da DPDAs und DFAs abgeschlossen unter Komplementbildung, Schnittbildung zwischen DPDA und DFA durch DPDA konstruierbar ist und Leerheitsproblem für CFLs entscheidbar ist
- $\overline{L_i} \cap L_j = \emptyset$ impliziert $L_i \subseteq L_j$
- Daher ist $\bigwedge_{(i,j) \in \{(1,2), (2,1)\}} \overline{L_i} \cap L_j = \emptyset$ äquivalent zu $L_1 = L_2$.

- CYK-Algorithmus: $w \in L(G)$ in $\mathcal{O}(n^3)$ für CFGs G in Chomsky-NF entscheiden.
- Kellerautomaten erkennen genau die CFLs
- Deterministisch kontextfreie Sprachen (DPDAs)
- Entscheidbarkeitsresultate für kontextfreie Sprachen