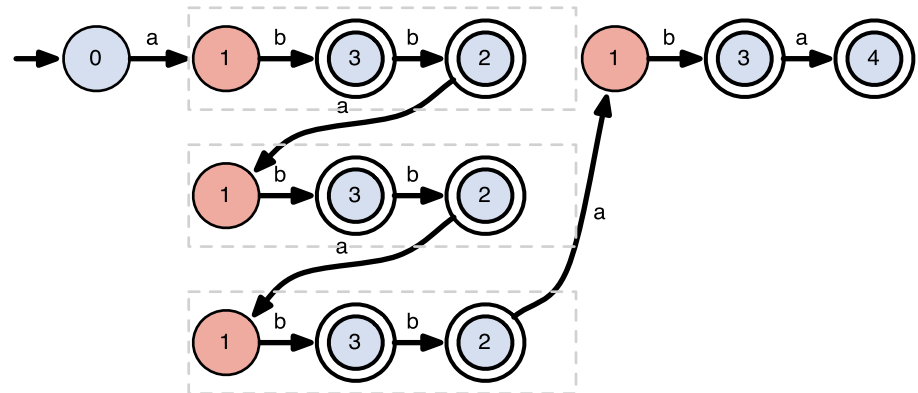


Berechenbarkeitstheorie

9. Vorlesung



Dr. Franziska Jahnke

Institut für Mathematische Logik und Grundlagenforschung

WWU Münster

CYK Algorithmus

CYK Algorithmus

- für ein Wort w bezeichnet $w[i, j]$ das Teilwort vom i ten bis j ten Zeichen

CYK Algorithmus

- für ein Wort w bezeichnet $w[i, j]$ das Teilwort vom i ten bis j ten Zeichen

Bsp.: $w = \text{autobahn} \rightarrow w[2, 5] = \text{utob}$

CYK Algorithmus

- für ein Wort w bezeichnet $w[i, j]$ das Teilwort vom i ten bis j ten Zeichen

Bsp.: $w = \text{autobahn} \rightarrow w[2, 5] = \text{utob}$

- $V_{ij} := \{U \in V \mid U \Rightarrow^* w[i, j]\}$

CYK Algorithmus

- für ein Wort w bezeichnet $w[i, j]$ das Teilwort vom i ten bis j ten Zeichen

Bsp.: $w = \text{autobahn} \rightarrow w[2, 5] = \text{utob}$

- $V_{ij} := \{U \in V \mid U \Rightarrow^* w[i, j]\}$
- **Teilprobleme des DP:** Bestimme die Mengen V_{ij}

CYK Algorithmus

- für ein Wort w bezeichnet $w[i, j]$ das Teilwort vom i ten bis j ten Zeichen

Bsp.: $w = \text{autobahn} \rightarrow w[2, 5] = \text{utob}$

- $V_{ij} := \{U \in V \mid U \Rightarrow^* w[i, j]\}$
- **Teilprobleme des DP:** Bestimme die Mengen V_{ij}
- $w \in L \iff S \in V_{1,n}$, mit $|w| = n$

CYK Algorithmus

- für ein Wort w bezeichnet $w[i, j]$ das Teilwort vom i ten bis j ten Zeichen

Bsp.: $w = \text{autobahn} \rightarrow w[2, 5] = \text{utob}$

- $V_{ij} := \{U \in V \mid U \Rightarrow^* w[i, j]\}$
- **Teilprobleme des DP:** Bestimme die Mengen V_{ij}
- $w \in L \iff S \in V_{1,n}$, mit $|w| = n$
- Initialisierung: $V_{ii} = \{X \mid X \rightarrow w[i, i]\}$ für $i = 1, \dots, n$

CYK Algorithmus

- für ein Wort w bezeichnet $w[i, j]$ das Teilwort vom i ten bis j ten Zeichen

Bsp.: $w = \text{autobahn} \rightarrow w[2, 5] = \text{utob}$

- $V_{ij} := \{U \in V \mid U \Rightarrow^* w[i, j]\}$
- **Teilprobleme des DP:** Bestimme die Mengen V_{ij}
- $w \in L \iff S \in V_{1,n}$, mit $|w| = n$
- Initialisierung: $V_{ii} = \{X \mid X \rightarrow w[i, i]\}$ für $i = 1, \dots, n$
- Rekursiver Schritt: Bestimme V_{ij} mit Hilfe bereits bekannter Teilprobleme

CYK Algorithmus

- für ein Wort w bezeichnet $w[i, j]$ das Teilwort vom i ten bis j ten Zeichen

Bsp.: $w = \text{autobahn} \rightarrow w[2, 5] = \text{utob}$

- $V_{ij} := \{U \in V \mid U \Rightarrow^* w[i, j]\}$
- **Teilprobleme des DP:** Bestimme die Mengen V_{ij}
- $w \in L \iff S \in V_{1,n}$, mit $|w| = n$
- Initialisierung: $V_{ii} = \{X \mid X \rightarrow w[i, i]\}$ für $i = 1, \dots, n$
- Rekursiver Schritt: Bestimme V_{ij} mit Hilfe bereits bekannter Teilprobleme
- Abstand zwischen i und j wird in jeder *Runde* um 1 größer

Finden einer Rekursion

Finden einer Rekursion

- sei $w[i, j] = z$, wir suchen Variablen U mit
 $U \Rightarrow XY \Rightarrow \dots \Rightarrow z$

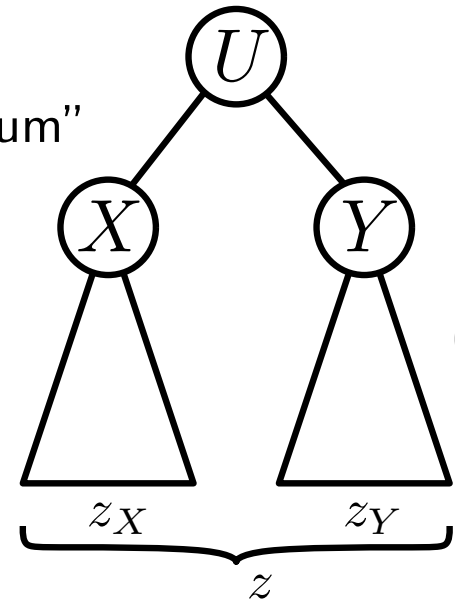
Finden einer Rekursion

- sei $w[i, j] = z$, wir suchen Variablen U mit

$$U \Rightarrow XY \Rightarrow \dots \Rightarrow z$$

2 Variablen da CNF

"Ableitungsbaum"



Finden einer Rekursion

- sei $w[i, j] = z$, wir suchen Variablen U mit

$$U \Rightarrow XY \Rightarrow \dots \Rightarrow z$$

2 Variablen da CNF

- es muss möglich sein $z = z_1 z_2$ zu

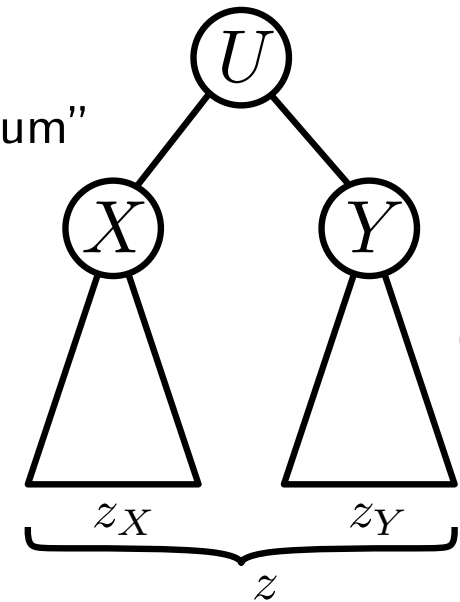
zerteilen mit

1. $U \Rightarrow U_1 U_2$

2. $U_1 \Rightarrow^* z_1$

3. $U_2 \Rightarrow^* z_2$

"Ableitungsbaum"



Finden einer Rekursion

- sei $w[i, j] = z$, wir suchen Variablen U mit

$$U \Rightarrow XY \Rightarrow \dots \Rightarrow z$$

"Ableitungsbaum"

2 Variablen da CNF

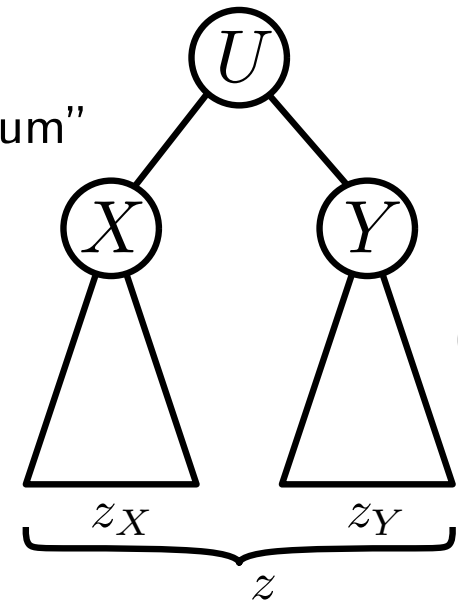
- es muss möglich sein $z = z_1 z_2$ zu zerteilen mit

1. $U \Rightarrow U_1 U_2$

2. $U_1 \Rightarrow^* z_1$

3. $U_2 \Rightarrow^* z_2$

} Teilprobleme schon gelöst



Finden einer Rekursion

- sei $w[i, j] = z$, wir suchen Variablen U mit

$$U \Rightarrow XY \Rightarrow \dots \Rightarrow z$$

"Ableitungsbaum"

2 Variablen da CNF

- es muss möglich sein $z = z_1 z_2$ zu zerteilen mit

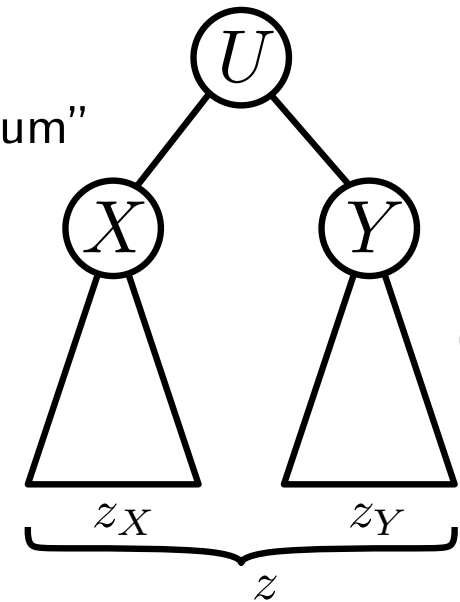
1. $U \Rightarrow U_1 U_2$

2. $U_1 \Rightarrow^* z_1$

3. $U_2 \Rightarrow^* z_2$

} Teilprobleme schon gelöst

- ich darf keine Zerteilung ausschließen \rightarrow ich muss alle Möglichkeiten z zu teilen in Betracht ziehen



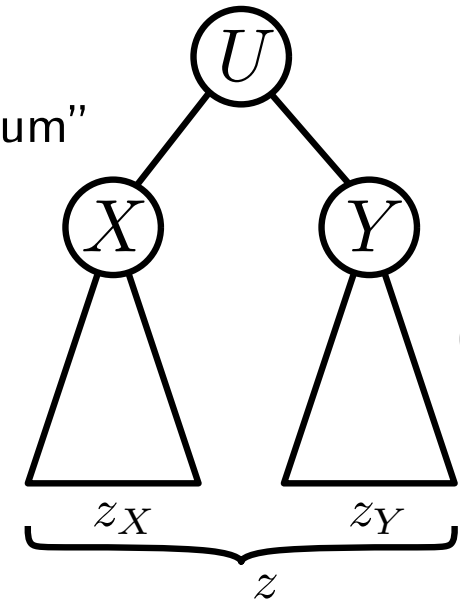
Finden einer Rekursion

- sei $w[i, j] = z$, wir suchen Variablen U mit

$$U \Rightarrow XY \Rightarrow \dots \Rightarrow z$$

"Ableitungsbaum"

2 Variablen da CNF



- es muss möglich sein $z = z_1 z_2$ zu zerteilen mit

1. $U \Rightarrow U_1 U_2$

2. $U_1 \Rightarrow^* z_1$

3. $U_2 \Rightarrow^* z_2$

} Teilprobleme schon gelöst

- ich darf keine Zerteilung ausschließen \rightarrow ich muss alle Möglichkeiten z zu teilen in Betracht ziehen

- **Rekursion:**

$$V_{ij} = \bigcup_{k=i}^{j-1} \{X \mid X \rightarrow AB \text{ mit } A \in V_{i,k}, B \in V_{k+1,j}\}$$

Algorithmus für $w \neq \varepsilon$

Algorithmus für $w \neq \varepsilon$

Datenstruktur: T , 2d Array, $T[i, j]$ enthält V_{ij}

Algorithmus für $w \neq \varepsilon$

Datenstruktur: T , 2d Array, $T[i, j]$ enthält V_{ij}

Algorithm 2: CYK Algorithmus

```
1 for  $i = 1$  to  $n$  do  $T[i, i] = \{X \in V \mid X \rightarrow w[i, i]\}$  ;
2 for  $d = 1$  to  $n - 1$  do
3   for  $i = 1$  to  $n - d$  do
4      $j = i + d$ ;
5     for  $t = i$  to  $j - 1$  do
6       for all  $A \rightarrow XY \in R$  do
7         if  $X \in T[i, t]$  and  $Y \in T[t + 1, j]$  then
8            $T[i, j] = T[i, j] \cup \{A\}$ ;
9         end
10      end
11    end
12  if  $S \in T[1, n]$  then return true;
13 else return false;
```

Algorithmus für $w \neq \varepsilon$

Datenstruktur: T , 2d Array, $T[i, j]$ enthält V_{ij}

Algorithm 2: CYK Algorithmus

```
1 for  $i = 1$  to  $n$  do  $T[i, i] = \{X \in V \mid X \rightarrow w[i, i]\}$  ; Initialisierung
2 for  $d = 1$  to  $n - 1$  do  $d$  ist Abstand  $i/j$ 
3   for  $i = 1$  to  $n - d$  do aktueller Eintrag  $T[i, j]$ 
4      $j = i + d$ ;
5     for  $t = i$  to  $j - 1$  do  $t$  ist Trennstelle
6       for all  $A \rightarrow XY \in R$  do
7         if  $X \in T[i, t]$  and  $Y \in T[t + 1, j]$  then
8            $T[i, j] = T[i, j] \cup \{A\}$ ;
9         end
10      end
11 end
12 if  $S \in T[1, n]$  then return true;
13 else return false;
```

Bsp. 1

$$\begin{aligned} S &\rightarrow AB \mid CA \\ A &\rightarrow AA \mid CB \mid a \\ B &\rightarrow AC \mid b \\ C &\rightarrow c \end{aligned}$$

$$w = cbaac$$

Bsp. 1

$S \rightarrow AB \mid CA$ $w = cbaac$
 $A \rightarrow AA \mid CB \mid a$
 $B \rightarrow AC \mid b$
 $C \rightarrow c$

	1	2	3	4	5
5					
4					
3					
2					
1					

Bsp. 1

$S \rightarrow AB \mid CA$ $w = cbaac$
 $A \rightarrow AA \mid CB \mid a$
 $B \rightarrow AC \mid b$
 $C \rightarrow c$

	1	2	3	4	5
5					C
4				A	
3			A		
2		B			
1	C				

Bsp. 1

$S \rightarrow AB \mid CA$ $w = cbaac$
 $A \rightarrow AA \mid CB \mid a$
 $B \rightarrow AC \mid b$
 $C \rightarrow c$

	1	2	3	4	5
5					C
4				A	
3			A		
2		B			
1	C	A			

Orange arrows indicate transitions: from C at (1,1) to A at (1,2), and from B at (2,2) to A at (1,2).

Bsp. 1

$S \rightarrow AB \mid CA$ $w = cbaac$
 $A \rightarrow AA \mid CB \mid a$
 $B \rightarrow AC \mid b$
 $C \rightarrow c$

	1	2	3	4	5
5					C
4				A	
3			A		
2		B	-		
1	C	A			

Bsp. 1

$S \rightarrow AB \mid CA$ $w = cbaac$
 $A \rightarrow AA \mid CB \mid a$
 $B \rightarrow AC \mid b$
 $C \rightarrow c$

	1	2	3	4	5
5					C
4				A	B
3			A	A	
2		B	-		
1	C	A			

Bsp. 1

$S \rightarrow AB \mid CA$ $w = cbaac$
 $A \rightarrow AA \mid CB \mid a$
 $B \rightarrow AC \mid b$
 $C \rightarrow c$

	1	2	3	4	5
5					C
4				A	B
3			A	A	
2		B	-		
1	C	A			

Bsp. 1

$S \rightarrow AB \mid CA$ $w = cbaac$
 $A \rightarrow AA \mid CB \mid a$
 $B \rightarrow AC \mid b$
 $C \rightarrow c$

	1	2	3	4	5
5					C
4				A	B
3			A	A	
2		B	-		
1	C	A	A		

Bsp. 1

$S \rightarrow AB \mid CA$ $w = cbaac$
 $A \rightarrow AA \mid CB \mid a$
 $B \rightarrow AC \mid b$
 $C \rightarrow c$

	1	2	3	4	5
5					C
4				A	B
3			A	A	S, B
2		B	-	-	
1	C	A	A		

Bsp. 1

$S \rightarrow AB \mid CA$ $w = cbaac$
 $A \rightarrow AA \mid CB \mid a$
 $B \rightarrow AC \mid b$
 $C \rightarrow c$

	1	2	3	4	5
5					C
4				A	B
3			A	A	S, B
2		B	-	-	
1	C	A	A		

Bsp. 1

$S \rightarrow AB \mid CA$ $w = cbaac$
 $A \rightarrow AA \mid CB \mid a$
 $B \rightarrow AC \mid b$
 $C \rightarrow c$

	1	2	3	4	5
5					C
4				A	B
3			A	A	S, B
2		B	-	-	
1	C	A	A		

Bsp. 1

$S \rightarrow AB \mid CA$ $w = cbaac$
 $A \rightarrow AA \mid CB \mid a$
 $B \rightarrow AC \mid b$
 $C \rightarrow c$

	1	2	3	4	5
5					C
4				A	B
3			A	A	S, B
2		B	-	-	
1	C	A	A	A	

Bsp. 1

$S \rightarrow AB \mid CA$ $w = cbaac$
 $A \rightarrow AA \mid CB \mid a$
 $B \rightarrow AC \mid b$
 $C \rightarrow c$

	1	2	3	4	5
5					C
4				A	B
3			A	A	S, B
2		B	-	-	-
1	C	A	A	A	

Bsp. 1

$S \rightarrow AB \mid CA$ $w = cbaac$
 $A \rightarrow AA \mid CB \mid a$
 $B \rightarrow AC \mid b$
 $C \rightarrow c$

	1	2	3	4	5
5					C
4				A	B
3			A	A	S, B
2		B	-	-	-
1	C	A	A	A	-

Bsp. 1

$S \rightarrow AB \mid CA$ $w = cbaac$
 $A \rightarrow AA \mid CB \mid a$
 $B \rightarrow AC \mid b$
 $C \rightarrow c$

	1	2	3	4	5
5					C
4				A	B
3			A	A	S, B
2		B	-	-	-
1	C	A	A	A	S

Bsp. 1

$S \rightarrow AB \mid CA$ $w = cbaac$
 $A \rightarrow AA \mid CB \mid a$
 $B \rightarrow AC \mid b$
 $C \rightarrow c$

	1	2	3	4	5
5					C
4				A	B
3			A	A	S, B
2		B	-	-	-
1	C	A	A	A	S

Bsp. 1

$S \rightarrow AB \mid CA$ $w = cbaac$
 $A \rightarrow AA \mid CB \mid a$
 $B \rightarrow AC \mid b$
 $C \rightarrow c$

	1	2	3	4	5
5					C
4				A	B
3			A	A	S, B
2		B	-	-	-
1	C	A	A	A	S, B

Bsp. 1

$$\begin{array}{l}
 S \rightarrow AB \mid CA \quad w = cbaac \\
 A \rightarrow AA \mid CB \mid a \\
 B \rightarrow AC \mid b \\
 C \rightarrow c
 \end{array}$$

	1	2	3	4	5
5					C
4				A	B
3			A	A	S, B
2		B	-	-	-
1	C	A	A	A	<u>S, B</u>

$S \in T[1, n]$ deshalb $w \in L(G)$

Bsp. 2

$S \rightarrow AB \mid CA$ $w = \text{aacaa}$
 $A \rightarrow AA \mid CB \mid a$
 $B \rightarrow AC \mid b$
 $C \rightarrow c$

	1	2	3	4	5
5					
4					
3					
2					
1					

Bsp. 2

$S \rightarrow AB \mid CA$ $w = \text{aacaa}$
 $A \rightarrow AA \mid CB \mid a$
 $B \rightarrow AC \mid b$
 $C \rightarrow c$

	1	2	3	4	5
5					A
4				A	
3			C		
2		A			
1	A				

Bsp. 2

$S \rightarrow AB \mid CA$ $w = \text{aacaa}$
 $A \rightarrow AA \mid CB \mid a$
 $B \rightarrow AC \mid b$
 $C \rightarrow c$

	1	2	3	4	5
5					A
4				A	
3			C		
2		A			
1	A	A			

Bsp. 2

$S \rightarrow AB \mid CA$ $w = \text{aacaa}$
 $A \rightarrow AA \mid CB \mid a$
 $B \rightarrow AC \mid b$
 $C \rightarrow c$

	1	2	3	4	5
5					A
4				A	
3			C		
2		A	B		
1	A	A			

Bsp. 2

$S \rightarrow AB \mid CA$ $w = \text{aaca}$
 $A \rightarrow AA \mid CB \mid a$
 $B \rightarrow AC \mid b$
 $C \rightarrow c$

	1	2	3	4	5
5					A
4				A	
3			C	S	
2		A	B		
1	A	A			

Bsp. 2

$S \rightarrow AB \mid CA$ $w = \text{aacaa}$
 $A \rightarrow AA \mid CB \mid a$
 $B \rightarrow AC \mid b$
 $C \rightarrow c$

	1	2	3	4	5
5					A
4				A	A
3			C	S	
2		A	B		
1	A	A			

Bsp. 2

$S \rightarrow AB \mid CA$ $w = \text{aacaa}$
 $A \rightarrow AA \mid CB \mid a$
 $B \rightarrow AC \mid b$
 $C \rightarrow c$

	1	2	3	4	5
5					A
4				A	A
3			C	S	
2		A	B		
1	A	A	S, B		

Bsp. 2

$S \rightarrow AB \mid CA$ $w = \text{aacaa}$
 $A \rightarrow AA \mid CB \mid a$
 $B \rightarrow AC \mid b$
 $C \rightarrow c$

	1	2	3	4	5
5					A
4				A	A
3			C	S	
2		A	B	-	
1	A	A	S, B		

Bsp. 2

$S \rightarrow AB \mid CA$ $w = \text{aacaa}$
 $A \rightarrow AA \mid CB \mid a$
 $B \rightarrow AC \mid b$
 $C \rightarrow c$

	1	2	3	4	5
5					A
4				A	A
3			C	S	S
2		A	B	-	
1	A	A	S, B		

Bsp. 2

$S \rightarrow AB \mid CA$ $w = \text{aacaa}$
 $A \rightarrow AA \mid CB \mid a$
 $B \rightarrow AC \mid b$
 $C \rightarrow c$

	1	2	3	4	5
5					A
4				A	A
3			C	S	S
2		A	B	–	
1	A	A	S, B	–	

Bsp. 2

$S \rightarrow AB \mid CA$ $w = \text{aacaa}$
 $A \rightarrow AA \mid CB \mid a$
 $B \rightarrow AC \mid b$
 $C \rightarrow c$

	1	2	3	4	5
5					A
4				A	A
3			C	S	S
2		A	B	-	-
1	A	A	S, B	-	

Bsp. 2

$S \rightarrow AB \mid CA$ $w = \text{aacaa}$
 $A \rightarrow AA \mid CB \mid a$
 $B \rightarrow AC \mid b$
 $C \rightarrow c$

	1	2	3	4	5
5					A
4				A	A
3			C	S	S
2		A	B	-	-
1	A	A	S, B	-	-

Bsp. 2

$$\begin{array}{l}
 S \rightarrow AB \mid CA \qquad w = \text{aacaa} \\
 A \rightarrow AA \mid CB \mid a \\
 B \rightarrow AC \mid b \\
 C \rightarrow c
 \end{array}$$

	1	2	3	4	5
5					A
4				A	A
3			C	S	S
2		A	B	-	-
1	A	A	S, B	-	-

$S \notin T[1, n]$ deshalb $w \notin L(G)$

Kellerautomaten

Kellerautomaten

- **Ziel:** Erweitere NEA Modell, sodass kontextfreie Sprachen erkannt werden können

Kellerautomaten

- **Ziel:** Erweitere NEA Modell, sodass kontextfreie Sprachen erkannt werden können
- **Idee:** NEA wird mit zusätzlichem Keller/Stack ausgestattet

Kellerautomaten

- **Ziel:** Erweitere NEA Modell, sodass kontextfreie Sprachen erkannt werden können
- **Idee:** NEA wird mit zusätzlichem Keller/Stack ausgestattet
 - Kellerinhalt ist Wort über **Kelleralphabet** (vertikal)

Kellerautomaten

- **Ziel:** Erweitere NEA Modell, sodass kontextfreie Sprachen erkannt werden können
- **Idee:** NEA wird mit zusätzlichem Keller/Stack ausgestattet
 - Kellerinhalt ist Wort über **Kelleralphabet** (vertikal)
 - oberstes Symbol auf dem Keller heißt **Top-Symbol**

Kellerautomaten

- **Ziel:** Erweitere NEA Modell, sodass kontextfreie Sprachen erkannt werden können
- **Idee:** NEA wird mit zusätzlichem Keller/Stack ausgestattet
 - Kellerinhalt ist Wort über **Kelleralphabet** (vertikal)
 - oberstes Symbol auf dem Keller heißt **Top-Symbol**
 - Zustandsübergänge hängen vom Top-Symbol ab

Kellerautomaten

- **Ziel:** Erweitere NEA Modell, sodass kontextfreie Sprachen erkannt werden können
- **Idee:** NEA wird mit zusätzlichem Keller/Stack ausgestattet
 - Kellerinhalt ist Wort über **Kelleralphabet** (vertikal)
 - oberstes Symbol auf dem Keller heißt **Top-Symbol**
 - Zustandsübergänge hängen vom Top-Symbol ab
 - Zustandsübergänge verändern Top-Symbol (Push, Pop, Push+Pop)

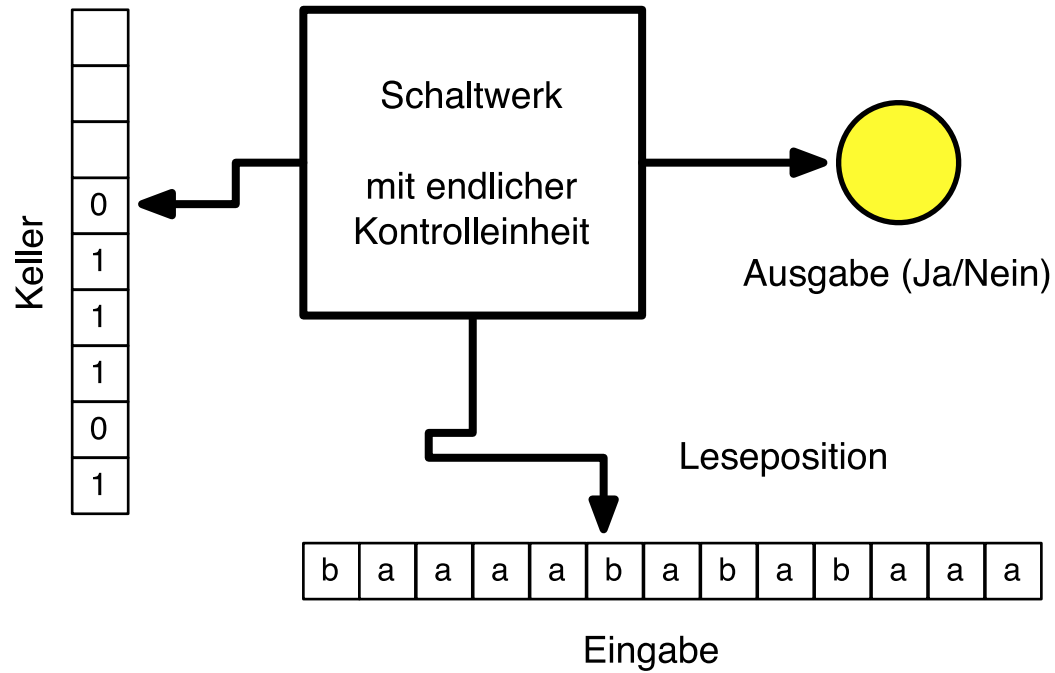
Kellerautomaten

- **Ziel:** Erweitere NEA Modell, sodass kontextfreie Sprachen erkannt werden können
- **Idee:** NEA wird mit zusätzlichem Keller/Stack ausgestattet
 - Kellerinhalt ist Wort über **Kelleralphabet** (vertikal)
 - oberstes Symbol auf dem Keller heißt **Top-Symbol**
 - Zustandsübergänge hängen vom Top-Symbol ab
 - Zustandsübergänge verändern Top-Symbol (Push, Pop, Push+Pop)
 - **Pop:** Top-Symbol wird entfernt
 - **Push:** Neues Top-Symbol wird *über* dem alten eingefügt
 - **Pop+Push:** Top-Symbol wird ausgetauscht

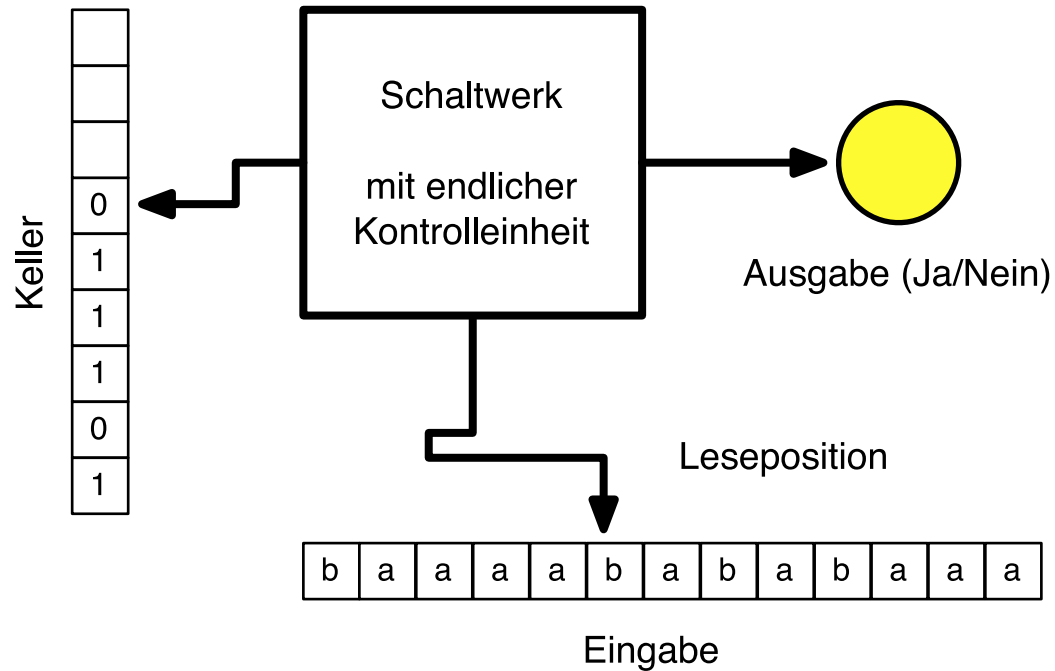
Kellerautomaten

- **Ziel:** Erweitere NEA Modell, sodass kontextfreie Sprachen erkannt werden können
- **Idee:** NEA wird mit zusätzlichem Keller/Stack ausgestattet
 - Kellerinhalt ist Wort über **Kelleralphabet** (vertikal)
 - oberstes Symbol auf dem Keller heißt **Top-Symbol**
 - Zustandsübergänge hängen vom Top-Symbol ab
 - Zustandsübergänge verändern Top-Symbol (Push, Pop, Push+Pop)
 - **Pop:** Top-Symbol wird entfernt
 - **Push:** Neues Top-Symbol wird *über* dem alten eingefügt
 - **Pop+Push:** Top-Symbol wird ausgetauscht
 - Nichtdeterminismus und ε -Übergänge bleiben erlaubt

Prinzip KA

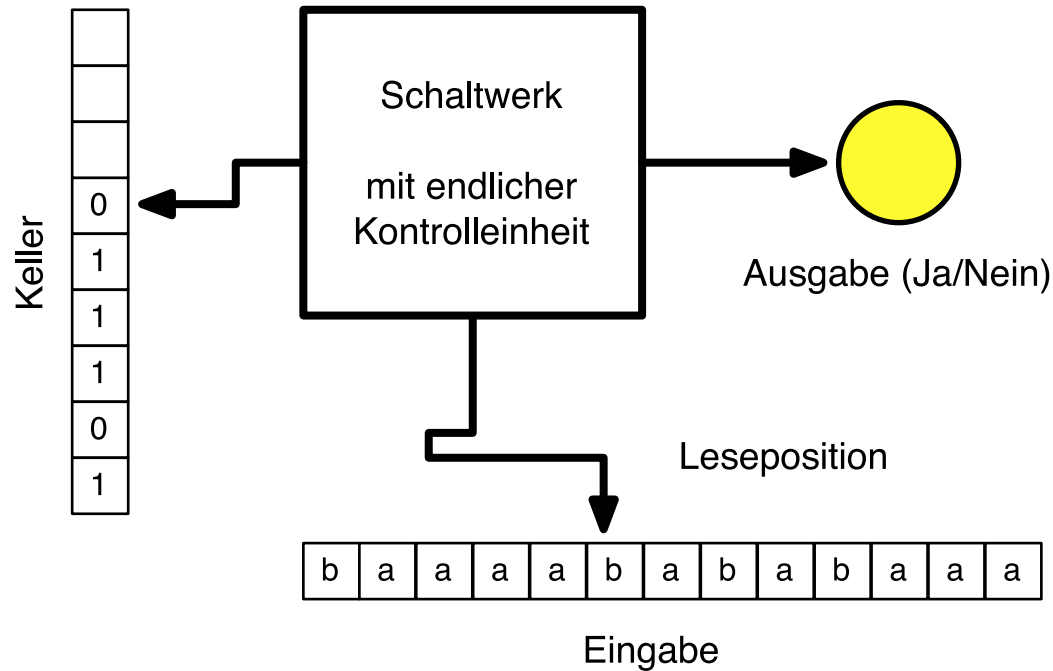


Prinzip KA



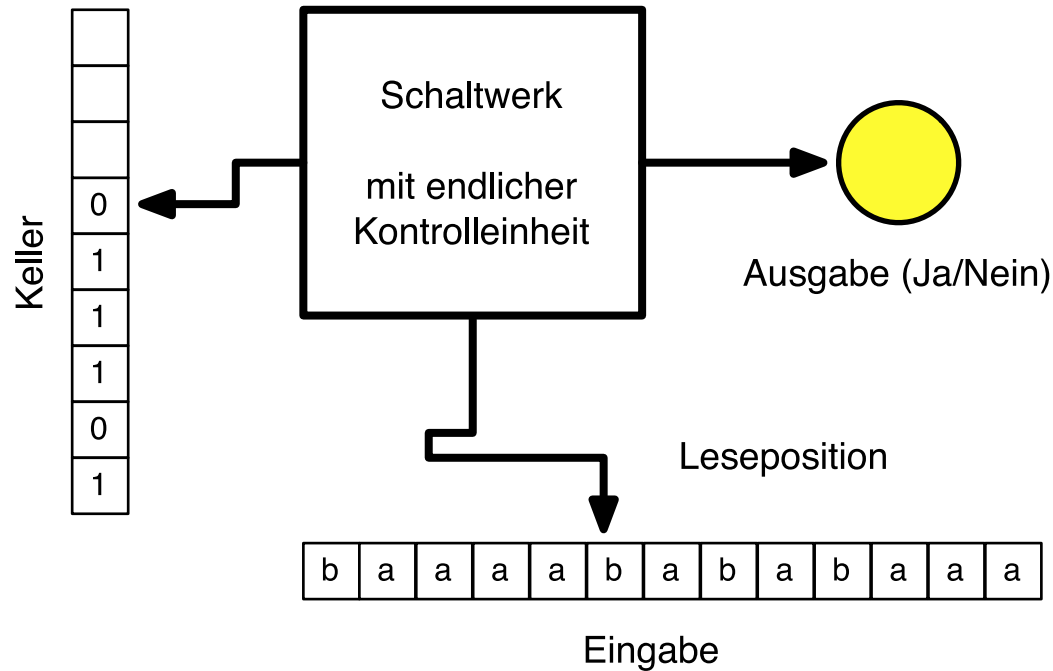
- nur von links nach rechts auf der Eingabe lesen

Prinzip KA



- nur von links nach rechts auf der Eingabe lesen
- nur oberstes Symbol des Kellers verfügbar

Prinzip KA



- nur von links nach rechts auf der Eingabe lesen
- nur oberstes Symbol des Kellers verfügbar
- nur ja/nein Antworten möglich

Definition

Ein **Kellerautomat (KA)** ist ein 6-Tupel

$(Q, \Sigma, \Gamma, \delta, q_0, F)$, mit:

- der Zustandsmenge Q ,
- dem Eingabealphabet Σ , dem Arbeitsalphabet Γ
- der Übergangsfunktion $\delta: Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \rightarrow \mathcal{P}(Q \times \Gamma_\varepsilon)$,
- $q_0 \in Q, F \subseteq Q$.

Hierbei: $\Sigma_\varepsilon := \Sigma \cup \{\varepsilon\}, \Gamma_\varepsilon := \Gamma \cup \{\varepsilon\}$

Definition

Ein **Kellerautomat (KA)** ist ein 6-Tupel

$(Q, \Sigma, \Gamma, \delta, q_0, F)$, mit:

- der Zustandsmenge Q ,
- dem Eingabealphabet Σ , dem Arbeitsalphabet Γ
- der Übergangsfunktion $\delta: Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \rightarrow \mathcal{P}(Q \times \Gamma_\varepsilon)$,
- $q_0 \in Q, F \subseteq Q$.

Hierbei: $\Sigma_\varepsilon := \Sigma \cup \{\varepsilon\}, \Gamma_\varepsilon := \Gamma \cup \{\varepsilon\}$

Interpretation Übergangsfunktion

Definition

Ein **Kellerautomat (KA)** ist ein 6-Tupel

$(Q, \Sigma, \Gamma, \delta, q_0, F)$, mit:

- der Zustandsmenge Q ,
- dem Eingabealphabet Σ , dem Arbeitsalphabet Γ
- der Übergangsfunktion $\delta: Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \rightarrow \mathcal{P}(Q \times \Gamma_\varepsilon)$,
- $q_0 \in Q, F \subseteq Q$.

Hierbei: $\Sigma_\varepsilon := \Sigma \cup \{\varepsilon\}, \Gamma_\varepsilon := \Gamma \cup \{\varepsilon\}$

Interpretation Übergangsfunktion

$$\delta(q_2, \mathbf{a}, \mathbf{x}) = \{(q_4, \mathbf{y})\}$$

Definition

Ein **Kellerautomat (KA)** ist ein 6-Tupel

$(Q, \Sigma, \Gamma, \delta, q_0, F)$, mit:

- der Zustandsmenge Q ,
- dem Eingabealphabet Σ , dem Arbeitsalphabet Γ
- der Übergangsfunktion $\delta: Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \rightarrow \mathcal{P}(Q \times \Gamma_\varepsilon)$,
- $q_0 \in Q, F \subseteq Q$.

Hierbei: $\Sigma_\varepsilon := \Sigma \cup \{\varepsilon\}, \Gamma_\varepsilon := \Gamma \cup \{\varepsilon\}$

Interpretation Übergangsfunktion

$$\delta(q_2, a, x) = \{(q_4, y)\}$$

Wenn Eingabezeichen a und Zustand q_2 und Top-Symbol x , gehe in Zustand q_4 und ersetze Topsymbol durch y

Definition

Ein **Kellerautomat (KA)** ist ein 6-Tupel

$(Q, \Sigma, \Gamma, \delta, q_0, F)$, mit:

- der Zustandsmenge Q ,
- dem Eingabealphabet Σ , dem Arbeitsalphabet Γ
- der Übergangsfunktion $\delta: Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \rightarrow \mathcal{P}(Q \times \Gamma_\varepsilon)$,
- $q_0 \in Q, F \subseteq Q$.

Hierbei: $\Sigma_\varepsilon := \Sigma \cup \{\varepsilon\}, \Gamma_\varepsilon := \Gamma \cup \{\varepsilon\}$

Interpretation Übergangsfunktion

$\delta(q_2, a, x) = \{(q_4, y)\}$ **Pop+Push**

Wenn Eingabezeichen a und Zustand q_2 und Top-Symbol x , gehe in Zustand q_4 und ersetze Topsymbol durch y

Definition

Ein **Kellerautomat (KA)** ist ein 6-Tupel

$(Q, \Sigma, \Gamma, \delta, q_0, F)$, mit:

- der Zustandsmenge Q ,
- dem Eingabealphabet Σ , dem Arbeitsalphabet Γ
- der Übergangsfunktion $\delta: Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \rightarrow \mathcal{P}(Q \times \Gamma_\varepsilon)$,
- $q_0 \in Q, F \subseteq Q$.

Hierbei: $\Sigma_\varepsilon := \Sigma \cup \{\varepsilon\}, \Gamma_\varepsilon := \Gamma \cup \{\varepsilon\}$

Interpretation Übergangsfunktion

$$\delta(q_2, a, x) = \{(q_4, y)\} \quad \text{Pop+Push}$$

Wenn Eingabezeichen a und Zustand q_2 und Top-Symbol x , gehe in Zustand q_4 und ersetze Topsymbol durch y

$$\delta(q_2, a, \varepsilon) = \{(q_4, y)\} \quad \text{Push}$$

Definition

Ein **Kellerautomat (KA)** ist ein 6-Tupel

$(Q, \Sigma, \Gamma, \delta, q_0, F)$, mit:

- der Zustandsmenge Q ,
- dem Eingabealphabet Σ , dem Arbeitsalphabet Γ
- der Übergangsfunktion $\delta: Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \rightarrow \mathcal{P}(Q \times \Gamma_\varepsilon)$,
- $q_0 \in Q, F \subseteq Q$.

Hierbei: $\Sigma_\varepsilon := \Sigma \cup \{\varepsilon\}, \Gamma_\varepsilon := \Gamma \cup \{\varepsilon\}$

Interpretation Übergangsfunktion

$\delta(q_2, a, x) = \{(q_4, y)\}$ **Pop+Push**

Wenn Eingabezeichen a und Zustand q_2 und Top-Symbol x , gehe in Zustand q_4 und ersetze Topsymbol durch y

$\delta(q_2, a, \varepsilon) = \{(q_4, y)\}$ **Push**

$\delta(q_2, a, x) = \{(q_4, \varepsilon)\}$ **Pop**

Akzeptanzbegriff KA

Akzeptanzbegriff KA

- **Konfiguration:** Element aus $Q \times \Gamma^*$ (aktueller Zustand + aktueller Kellerinhalt)

Akzeptanzbegriff KA

- **Konfiguration:** Element aus $Q \times \Gamma^*$ (aktueller Zustand + aktueller Kellerinhalt)
- **w -Lauf:** Sequenz von Konfigurationen $((p_1, s_1), (p_2, s_2), \dots, (p_{n+1}, s_{n+1}))$ mit

Akzeptanzbegriff KA

- **Konfiguration:** Element aus $Q \times \Gamma^*$ (aktueller Zustand + aktueller Kellerinhalt)
- **w -Lauf:** Sequenz von Konfigurationen $((p_1, s_1), (p_2, s_2), \dots, (p_{n+1}, s_{n+1}))$ mit
 1. $(p_1, s_1) = (q_0, \varepsilon)$
 2. $\forall i = 1, \dots, n: \delta(p_i, x_i, a) \ni (p_{i+1}, b)$ mit $w = x_1 x_2 \dots, x_i \in \Sigma_\varepsilon$ und $s_i = ta, s_{i+1} = tb$ wobei $t \in \Gamma^*$

Akzeptanzbegriff KA

- **Konfiguration:** Element aus $Q \times \Gamma^*$ (aktueller Zustand + aktueller Kellerinhalt)
- **w -Lauf:** Sequenz von Konfigurationen $((p_1, s_1), (p_2, s_2), \dots, (p_{n+1}, s_{n+1}))$ mit
 1. $(p_1, s_1) = (q_0, \varepsilon)$
 2. $\forall i = 1, \dots, n: \delta(p_i, x_i, a) \ni (p_{i+1}, b)$ mit $w = x_1 x_2 \dots, x_i \in \Sigma_\varepsilon$ und $s_i = ta, s_{i+1} = tb$ wobei $t \in \Gamma^*$

Definiton

Ein KA $K = (Q, \Sigma, \Gamma, \delta, q_0, F)$ akzeptiert ein Wort $w \in \Sigma^*$, gdw. ein w -Lauf existiert, der in einer Konfiguration mit einem Zustand aus F endet.

Die von K erkannte Sprache lautet

$$L(K) = \{w \in \Sigma^* \mid K \text{ erkennt } w\}$$

Bsp. Kellerautomat für $\{a^n b^n \mid n \geq 0\}$

Bsp. Kellerautomat für $\{a^n b^n \mid n \geq 0\}$

$$Q = \{q_0, q_1, q_2, q_3\}$$

$$\Sigma = \{a, b\}$$

$$\Gamma = \{\$, a\}$$

$$F = \{q_0, q_3\}$$

Bsp. Kellerautomat für $\{a^n b^n \mid n \geq 0\}$

$$Q = \{q_0, q_1, q_2, q_3\}$$

$$\Sigma = \{a, b\}$$

$$\Gamma = \{\$, a\}$$

$$F = \{q_0, q_3\}$$

$$\delta(q_0, \varepsilon, \varepsilon) = \{(q_1, \$)\}$$

$$\delta(q_1, a, \varepsilon) = \{(q_1, a)\}$$

$$\delta(q_1, b, a) = \{(q_2, \varepsilon)\}$$

$$\delta(q_2, b, a) = \{(q_2, \varepsilon)\}$$

$$\delta(q_2, \varepsilon, \$) = \{(q_3, \varepsilon)\}$$

Bsp. Kellerautomat für $\{a^n b^n \mid n \geq 0\}$

$$Q = \{q_0, q_1, q_2, q_3\}$$

$$\Sigma = \{a, b\}$$

$$\Gamma = \{\$, a\}$$

$$F = \{q_0, q_3\}$$

$\delta(q_0, \varepsilon, \varepsilon)$	$=$	$\{(q_1, \$)\}$	Push \$
$\delta(q_1, a, \varepsilon)$	$=$	$\{(q_1, a)\}$	Push a
$\delta(q_1, b, a)$	$=$	$\{(q_2, \varepsilon)\}$	Pop a
$\delta(q_2, b, a)$	$=$	$\{(q_2, \varepsilon)\}$	Pop a
$\delta(q_2, \varepsilon, \$)$	$=$	$\{(q_3, \varepsilon)\}$	Pop \$

Bsp. Kellerautomat für $\{a^n b^n \mid n \geq 0\}$

$$Q = \{q_0, q_1, q_2, q_3\}$$

$$\Sigma = \{a, b\}$$

$$\Gamma = \{\$, a\}$$

$$F = \{q_0, q_3\}$$

$$\delta(q_0, \varepsilon, \varepsilon) = \{(q_1, \$)\} \quad \text{Push \$}$$

$$\delta(q_1, a, \varepsilon) = \{(q_1, a)\} \quad \text{Push a}$$

$$\delta(q_1, b, a) = \{(q_2, \varepsilon)\} \quad \text{Pop a}$$

$$\delta(q_2, b, a) = \{(q_2, \varepsilon)\} \quad \text{Pop a}$$

$$\delta(q_2, \varepsilon, \$) = \{(q_3, \varepsilon)\} \quad \text{Pop \$}$$

Diagrammdarstellung für $(q_j, b) \in \delta(q_i, x, a)$

Bsp. Kellerautomat für $\{a^n b^n \mid n \geq 0\}$

$$Q = \{q_0, q_1, q_2, q_3\}$$

$$\Sigma = \{a, b\}$$

$$\Gamma = \{\$, a\}$$

$$F = \{q_0, q_3\}$$

$$\delta(q_0, \varepsilon, \varepsilon) = \{(q_1, \$)\} \quad \text{Push \$}$$

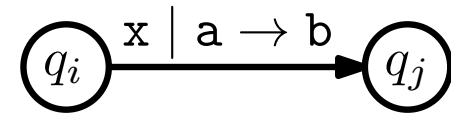
$$\delta(q_1, a, \varepsilon) = \{(q_1, a)\} \quad \text{Push a}$$

$$\delta(q_1, b, a) = \{(q_2, \varepsilon)\} \quad \text{Pop a}$$

$$\delta(q_2, b, a) = \{(q_2, \varepsilon)\} \quad \text{Pop a}$$

$$\delta(q_2, \varepsilon, \$) = \{(q_3, \varepsilon)\} \quad \text{Pop \$}$$

Diagrammdarstellung für $(q_j, b) \in \delta(q_i, x, a)$



Bsp. Kellerautomat für $\{a^n b^n \mid n \geq 0\}$

$$Q = \{q_0, q_1, q_2, q_3\}$$

$$\Sigma = \{a, b\}$$

$$\Gamma = \{\$, a\}$$

$$F = \{q_0, q_3\}$$

$$\delta(q_0, \varepsilon, \varepsilon) = \{(q_1, \$)\} \quad \text{Push \$}$$

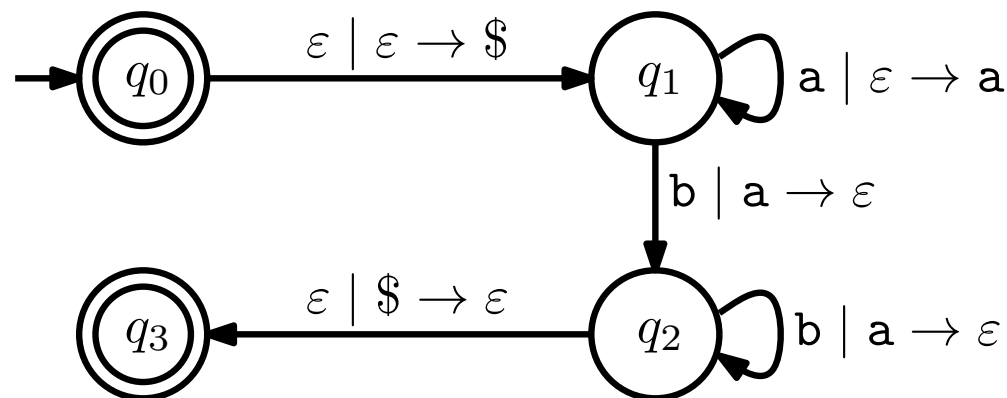
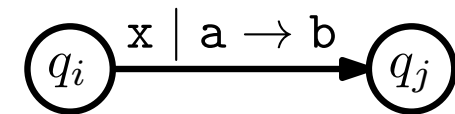
$$\delta(q_1, a, \varepsilon) = \{(q_1, a)\} \quad \text{Push a}$$

$$\delta(q_1, b, a) = \{(q_2, \varepsilon)\} \quad \text{Pop a}$$

$$\delta(q_2, b, a) = \{(q_2, \varepsilon)\} \quad \text{Pop a}$$

$$\delta(q_2, \varepsilon, \$) = \{(q_3, \varepsilon)\} \quad \text{Pop \$}$$

Diagrammdarstellung für $(q_j, b) \in \delta(q_i, x, a)$



Satz 12

Eine Sprache L wird genau dann von einem KA erkannt, wenn $L \in \text{CFL}$.

Satz 12

Eine Sprache L wird genau dann von einem KA erkannt, wenn $L \in \text{CFL}$.

1. Teil des Beweises

Wenn $L \in \text{CFL}$ dann gibt es KA $K = (Q, \Sigma, \Gamma, \delta, q_0, F)$ mit $L(K) = L$

Satz 12

Eine Sprache L wird genau dann von einem KA erkannt, wenn $L \in \text{CFL}$.

1. Teil des Beweises

Wenn $L \in \text{CFL}$ dann gibt es KA $K = (Q, \Sigma, \Gamma, \delta, q_0, F)$ mit $L(K) = L$

Ansatz

- $L \in \text{CFL}$ heißt, es gibt kfG $G = (V, \Sigma, R, S)$ mit $L(G) = L$

Satz 12

Eine Sprache L wird genau dann von einem KA erkannt, wenn $L \in \text{CFL}$.

1. Teil des Beweises

Wenn $L \in \text{CFL}$ dann gibt es KA $K = (Q, \Sigma, \Gamma, \delta, q_0, F)$ mit $L(K) = L$

Ansatz

- $L \in \text{CFL}$ heißt, es gibt kfG $G = (V, \Sigma, R, S)$ mit $L(G) = L$
- $w \in L \iff S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \dots \Rightarrow w$

Satz 12


Eine Sprache L wird genau dann von einem KA erkannt, wenn $L \in \text{CFL}$.

1. Teil des Beweises

Wenn $L \in \text{CFL}$ dann gibt es KA $K = (Q, \Sigma, \Gamma, \delta, q_0, F)$ mit $L(K) = L$

Ansatz

■ $L \in \text{CFL}$ heißt, es gibt kfG $G = (V, \Sigma, R, S)$ mit $L(G) = L$

■ $w \in L \iff S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \dots \Rightarrow w$

Simuliere Ableitungen durch KA

Satz 12

Eine Sprache L wird genau dann von einem KA erkannt, wenn $L \in \text{CFL}$.

1. Teil des Beweises

Wenn $L \in \text{CFL}$ dann gibt es KA $K = (Q, \Sigma, \Gamma, \delta, q_0, F)$ mit $L(K) = L$

Ansatz

- $L \in \text{CFL}$ heißt, es gibt kfG $G = (V, \Sigma, R, S)$ mit $L(G) = L$
- $w \in L \iff S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \dots \Rightarrow w$ Kellerinhalt = w_i
⏟
Simuliere Ableitungen durch KA

Satz 12

Eine Sprache L wird genau dann von einem KA erkannt, wenn $L \in \text{CFL}$.

1. Teil des Beweises

Wenn $L \in \text{CFL}$ dann gibt es KA $K = (Q, \Sigma, \Gamma, \delta, q_0, F)$ mit $L(K) = L$

Ansatz

- $L \in \text{CFL}$ heißt, es gibt kfG $G = (V, \Sigma, R, S)$ mit $L(G) = L$
- $w \in L \iff S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \dots \Rightarrow w$ Kellereinhalte = w_i
⏟
Simuliere Ableitungen durch KA
- Problem: nur das Top-Symbol kann ersetzt werden!

Satz 12

Eine Sprache L wird genau dann von einem KA erkannt, wenn $L \in \text{CFL}$.

1. Teil des Beweiss

Wenn $L \in \text{CFL}$ dann gibt es KA $K = (Q, \Sigma, \Gamma, \delta, q_0, F)$ mit $L(K) = L$

Ansatz

- $L \in \text{CFL}$ heißt, es gibt kfG $G = (V, \Sigma, R, S)$ mit $L(G) = L$
- $w \in L \iff S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \dots \Rightarrow w$ Kellereinhalte = w_i
⏟
Simuliere Ableitungen durch KA
- Problem: nur das Top-Symbol kann ersetzt werden!
- Lösung: Leite immer die am weitesten links stehende Variable ab

Satz 12


Eine Sprache L wird genau dann von einem KA erkannt, wenn $L \in \text{CFL}$.

1. Teil des Beweises

Wenn $L \in \text{CFL}$ dann gibt es KA $K = (Q, \Sigma, \Gamma, \delta, q_0, F)$ mit $L(K) = L$

Ansatz

- $L \in \text{CFL}$ heißt, es gibt kfG $G = (V, \Sigma, R, S)$ mit $L(G) = L$
 - $w \in L \iff S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \dots \Rightarrow w$ Kellereinhalte = w_i



Simuliere Ableitungen durch KA
 - Problem: nur das Top-Symbol kann ersetzt werden!
 - Lösung: Leite immer die am weitesten links stehende Variable ab
- Bsp: Kellereinhalte $w_i = a \ b \ A \ b \ A \ B \ c$


Satz 12

Eine Sprache L wird genau dann von einem KA erkannt, wenn $L \in \text{CFL}$.

1. Teil des Beweises

Wenn $L \in \text{CFL}$ dann gibt es KA $K = (Q, \Sigma, \Gamma, \delta, q_0, F)$ mit $L(K) = L$

Ansatz

- $L \in \text{CFL}$ heißt, es gibt kfG $G = (V, \Sigma, R, S)$ mit $L(G) = L$
 - $w \in L \iff S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \dots \Rightarrow w$ Kellereinhalte = w_i


Simuliere Ableitungen durch KA
 - Problem: nur das Top-Symbol kann ersetzt werden!
 - Lösung: Leite immer die am weitesten links stehende Variable ab
- Bsp: Kellereinhalte $w_i = a b \textcircled{A} b A B c$
- ← Hier Ableiten


Satz 12

Eine Sprache L wird genau dann von einem KA erkannt, wenn $L \in \text{CFL}$.

1. Teil des Beweises

Wenn $L \in \text{CFL}$ dann gibt es KA $K = (Q, \Sigma, \Gamma, \delta, q_0, F)$ mit $L(K) = L$

Ansatz

- $L \in \text{CFL}$ heißt, es gibt kfG $G = (V, \Sigma, R, S)$ mit $L(G) = L$
- $w \in L \iff S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \dots \Rightarrow w$ Kellerinhalt = w_i


- Problem: nur das Top-Symbol kann ersetzt werden!
- Lösung: Leite immer die am weitesten links stehende Variable ab

Bsp: Kellerinhalt $w_i = a b A b A B c$

Präfix aus Terminalsymbolen Hier Ableiten


Satz 12

Eine Sprache L wird genau dann von einem KA erkannt, wenn $L \in \text{CFL}$.

1. Teil des Beweiss

Wenn $L \in \text{CFL}$ dann gibt es KA $K = (Q, \Sigma, \Gamma, \delta, q_0, F)$ mit $L(K) = L$

Ansatz

- $L \in \text{CFL}$ heißt, es gibt kfG $G = (V, \Sigma, R, S)$ mit $L(G) = L$
- $w \in L \iff S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \dots \Rightarrow w$ Kellereinhalte = w_i


- Problem: nur das Top-Symbol kann ersetzt werden!
- Lösung: Leite immer die am weitesten links stehende Variable ab

Bsp: Kellereinhalte $w_i = a \ b \ A \ b \ A \ B \ c$

- 
 Präfix aus Terminalsymbolen
- 
 Hier Ableiten
- 
 Muss mit w übereinstimmen, kann also schon geprüft werden

Satz 12

Eine Sprache L wird genau dann von einem KA erkannt, wenn $L \in \text{CFL}$.

1. Teil des Beweiss

Wenn $L \in \text{CFL}$ dann gibt es KA $K = (Q, \Sigma, \Gamma, \delta, q_0, F)$ mit $L(K) = L$

Ansatz

- $L \in \text{CFL}$ heißt, es gibt kfG $G = (V, \Sigma, R, S)$ mit $L(G) = L$
- $w \in L \iff S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \dots \Rightarrow w$ Kellerinhalt = w_i


Simuliere Ableitungen durch KA

- Problem: nur das Top-Symbol kann ersetzt werden!
- Lösung: Leite immer die am weitesten links stehende Variable ab

Bsp: Kellerinhalt $w_i = a \underbrace{b} \underbrace{A} b A B c$

 Präfix aus Terminalsymbolen

 Hier Ableiten (Top-Symbol)

 Muss mit w übereinstimmen, kann also schon geprüft werden

Konstruktion des KA

- 4 Hauptzustände: $q_0, q_1, q_{\text{loop}}, q_2$

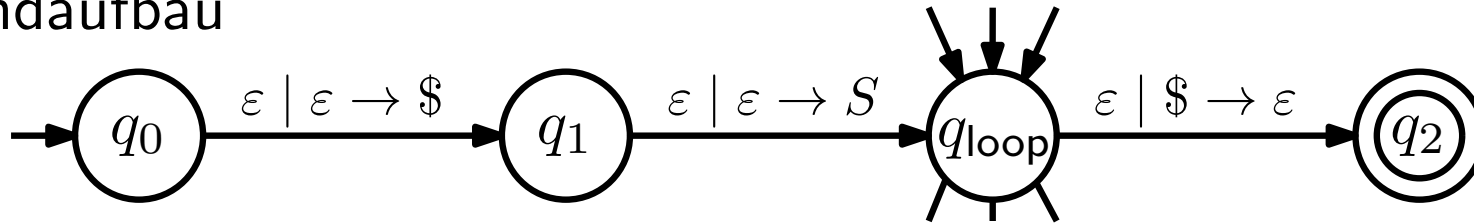
Konstruktion des KA

- 4 Hauptzustände: $q_0, q_1, q_{\text{loop}}, q_2$
- $F = \{q_2\}$, q_0 Startzustand, $\Gamma = \Sigma \cup V$

Konstruktion des KA

- 4 Hauptzustände: $q_0, q_1, q_{\text{loop}}, q_2$
- $F = \{q_2\}$, q_0 Startzustand, $\Gamma = \Sigma \cup V$

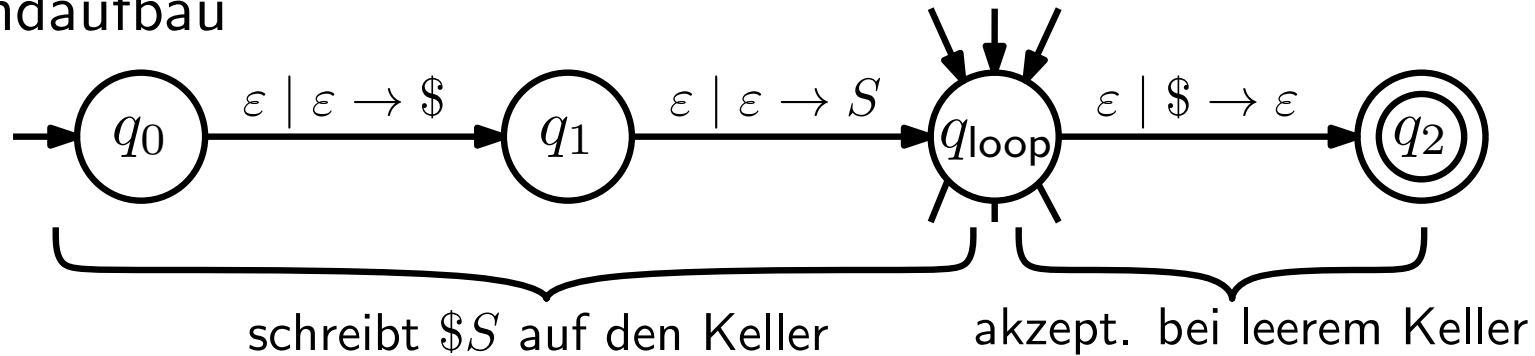
Grundaufbau



Konstruktion des KA

- 4 Hauptzustände: q_0, q_1, q_{loop}, q_2
- $F = \{q_2\}$, q_0 Startzustand, $\Gamma = \Sigma \cup V$

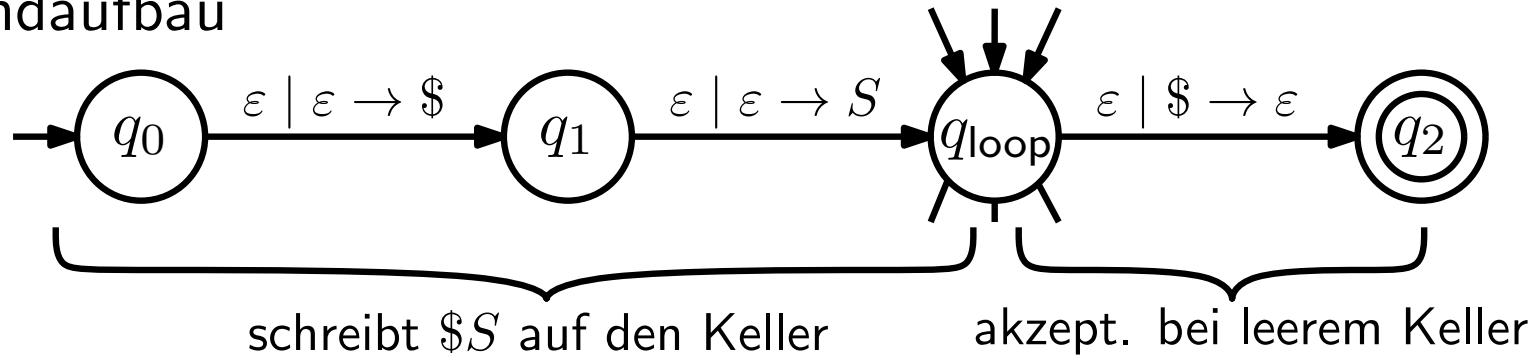
Grundaufbau



Konstruktion des KA

- 4 Hauptzustände: q_0, q_1, q_{loop}, q_2
- $F = \{q_2\}$, q_0 Startzustand, $\Gamma = \Sigma \cup V$

Grundaufbau

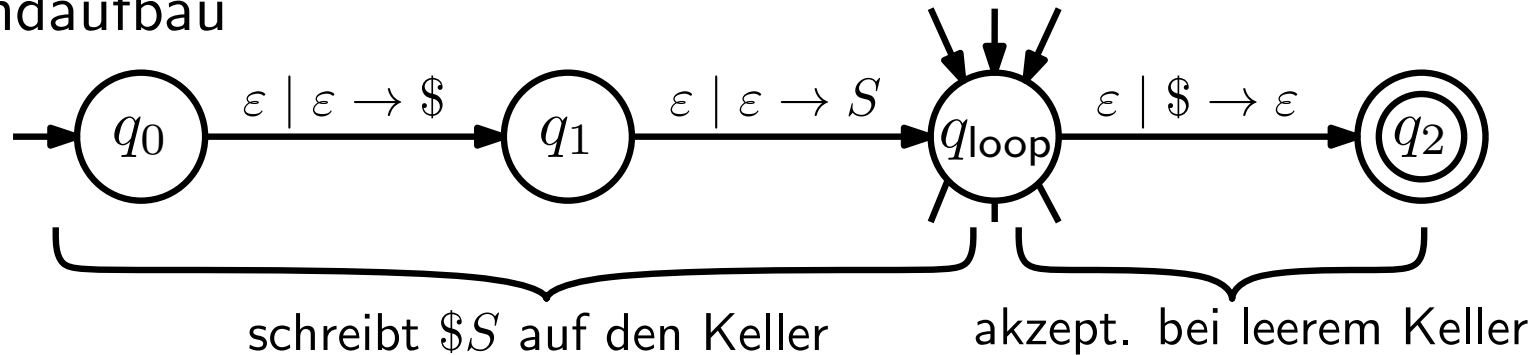


1. Erweiterung: Erlaubt Ableitungen des Top-Symbols

Konstruktion des KA

- 4 Hauptzustände: $q_0, q_1, q_{\text{loop}}, q_2$
- $F = \{q_2\}$, q_0 Startzustand, $\Gamma = \Sigma \cup V$

Grundaufbau



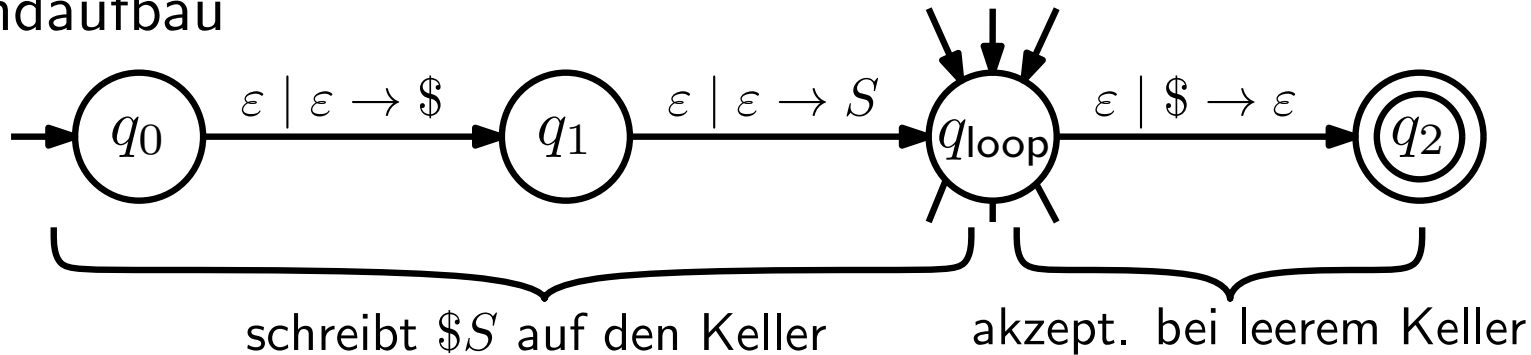
1. Erweiterung: Erlaubt Ableitungen des Top-Symbols

- für jede Regel $A \rightarrow z = z_1 z_2 \cdots z_k$ ($z_i \in \Sigma \cup V$) füge folgende Schleife hinzu

Konstruktion des KA

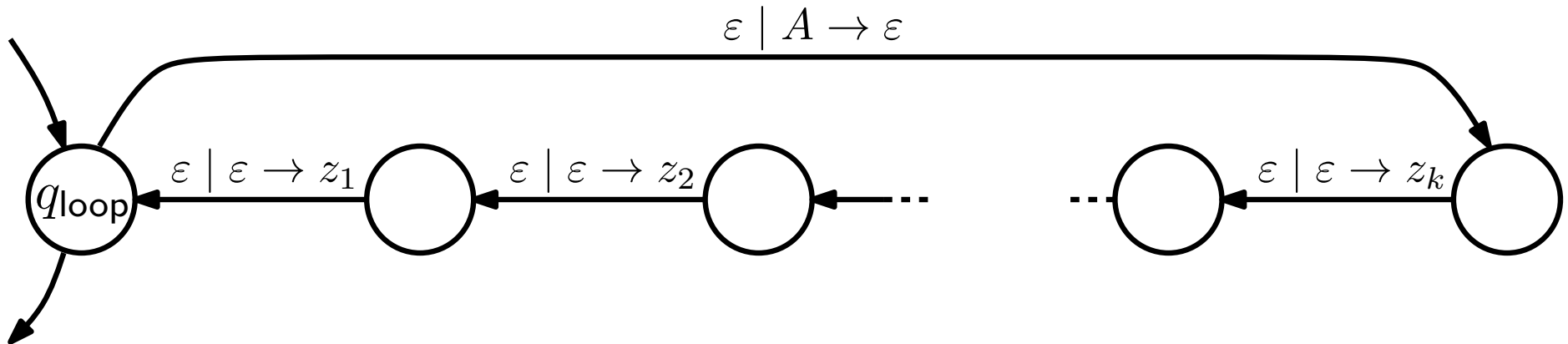
- 4 Hauptzustände: q_0, q_1, q_{loop}, q_2
- $F = \{q_2\}$, q_0 Startzustand, $\Gamma = \Sigma \cup V$

Grundaufbau



1. Erweiterung: Erlaubt Ableitungen des Top-Symbols

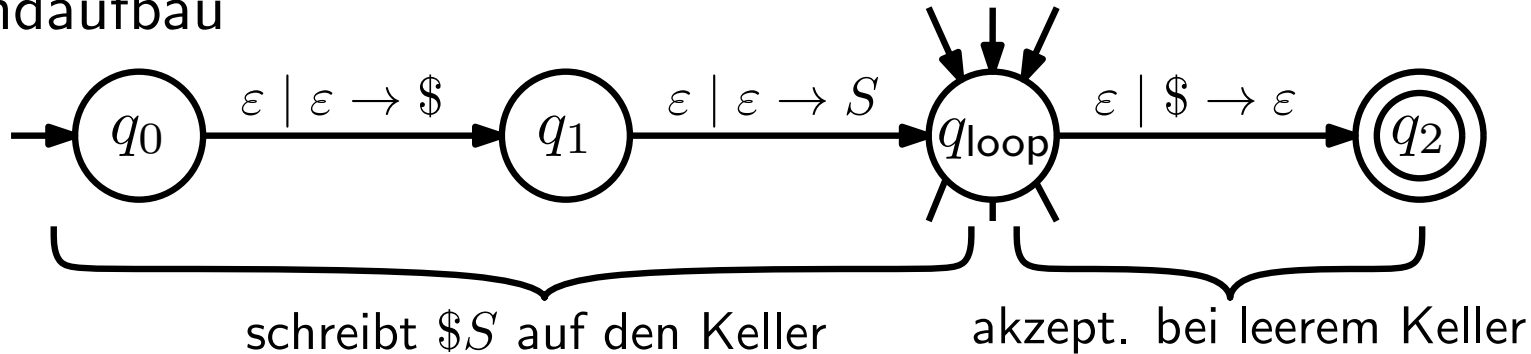
- für jede Regel $A \rightarrow z = z_1 z_2 \cdots z_k$ ($z_i \in \Sigma \cup V$) füge folgende Schleife hinzu



Konstruktion des KA

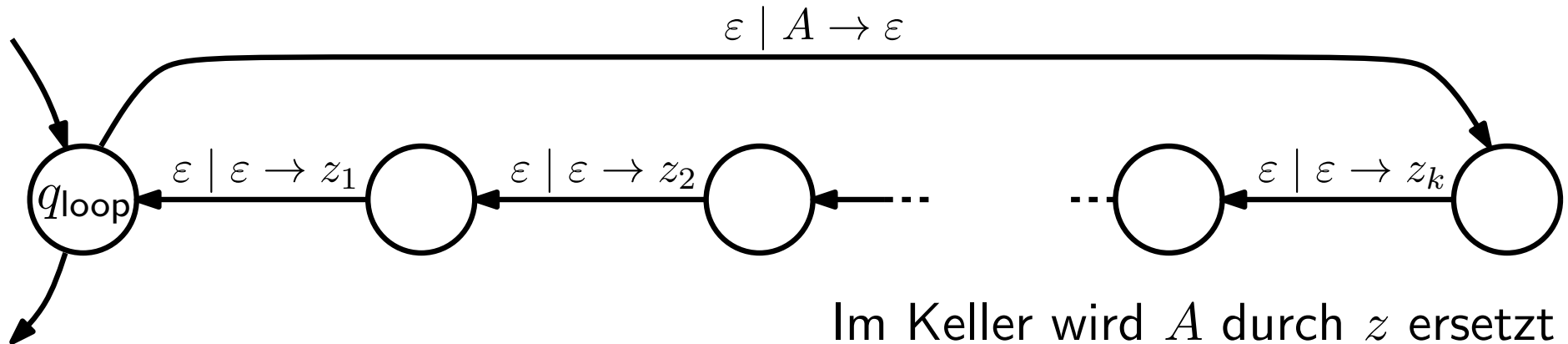
- 4 Hauptzustände: q_0, q_1, q_{loop}, q_2
- $F = \{q_2\}$, q_0 Startzustand, $\Gamma = \Sigma \cup V$

Grundaufbau



1. Erweiterung: Erlaubt Ableitungen des Top-Symbols

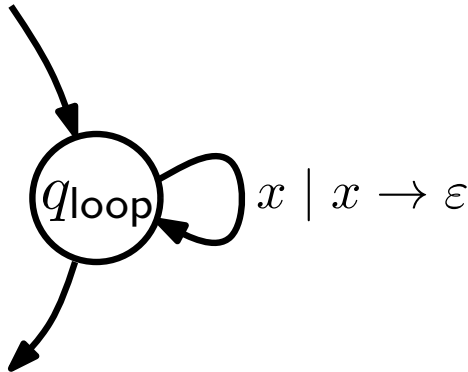
- für jede Regel $A \rightarrow z = z_1 z_2 \cdots z_k$ ($z_i \in \Sigma \cup V$) füge folgende Schleife hinzu



- wenn das Top-Symbol ein Terminal ist, muss es mit der Eingabe übereinstimmen

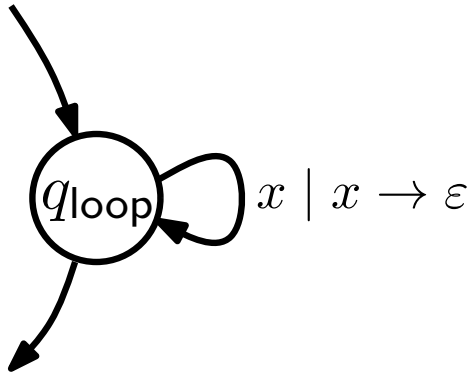
- wenn das Top-Symbol ein Terminal ist, muss es mit der Eingabe übereinstimmen

2. Erweiterung: Abgleichen des Top-Symbols mit der Eingabe



- wenn das Top-Symbol ein Terminal ist, muss es mit der Eingabe übereinstimmen

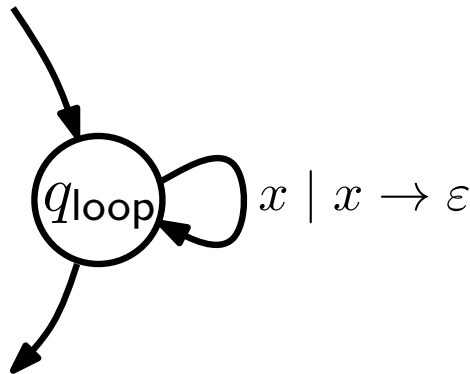
2. Erweiterung: Abgleichen des Top-Symbols mit der Eingabe



für jedes $x \in \Sigma$ wird so ein Übergang eingefügt

- wenn das Top-Symbol ein Terminal ist, muss es mit der Eingabe übereinstimmen

2. Erweiterung: Abgleichen des Top-Symbols mit der Eingabe



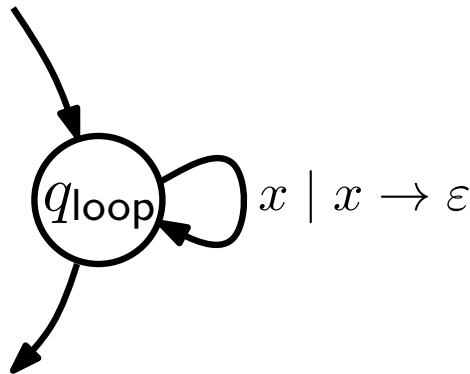
für jedes $x \in \Sigma$ wird so ein Übergang eingefügt

Zusammenfassung:

- KA bildet die Linksableitung von w nach (1. Erweiterung)

- wenn das Top-Symbol ein Terminal ist, muss es mit der Eingabe übereinstimmen

2. Erweiterung: Abgleichen des Top-Symbols mit der Eingabe



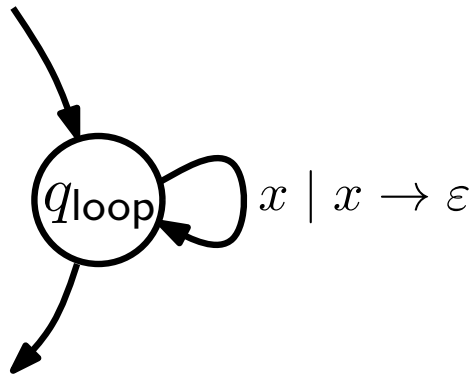
für jedes $x \in \Sigma$ wird so ein Übergang eingefügt

Zusammenfassung:

- KA bildet die Linksableitung von w nach (1. Erweiterung)
→ immer die am weitesten links stehende Variable ableiten

- wenn das Top-Symbol ein Terminal ist, muss es mit der Eingabe übereinstimmen

2. Erweiterung: Abgleichen des Top-Symbols mit der Eingabe



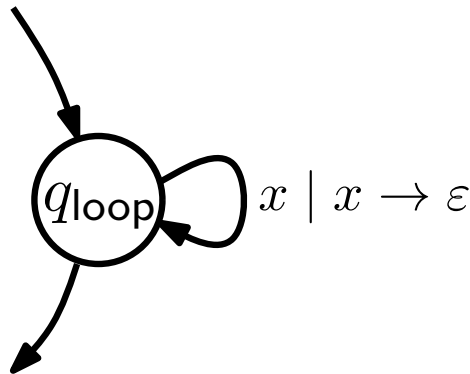
für jedes $x \in \Sigma$ wird so ein Übergang eingefügt

Zusammenfassung:

- KA bildet die Linksableitung von w nach (1. Erweiterung)
 - immer die am weitesten links stehende Variable ableiten
- stehen Terminalsymbole links, können diese mit der Eingabe abgeglichen werden (2. Erweiterung)

- wenn das Top-Symbol ein Terminal ist, muss es mit der Eingabe übereinstimmen

2. Erweiterung: Abgleichen des Top-Symbols mit der Eingabe



für jedes $x \in \Sigma$ wird so ein Übergang eingefügt

Zusammenfassung:

- KA bildet die Linksableitung von w nach (1. Erweiterung)
 - immer die am weitesten links stehende Variable ableiten
- stehen Terminalsymbole links, können diese mit der Eingabe abgeglichen werden (2. Erweiterung)
- Akzeptanz \iff alle Zeichen der Eingabe wurden abgeglichen, keine Zeichen verbleiben auf Keller

2. Teil des Beweises

Für jeden KA $K = (Q, \Sigma, \Gamma, \delta, q_0, F)$ gilt $L(K) \in \text{CFL}$

2. Teil des Beweises

Für jeden KA $K = (Q, \Sigma, \Gamma, \delta, q_0, F)$ gilt $L(K) \in \text{CFL}$

1. Schritt Modifikation des KA

2. Teil des Beweises

Für jeden KA $K = (Q, \Sigma, \Gamma, \delta, q_0, F)$ gilt $L(K) \in \text{CFL}$

1. Schritt Modifikation des KA

- Baue K so um, dass
 1. $F = \{q_F\}$
 2. akzeptiert wird nur mit leerem Keller
 3. nur Push oder Pop (kein Push+Pop)

2. Teil des Beweises

Für jeden KA $K = (Q, \Sigma, \Gamma, \delta, q_0, F)$ gilt $L(K) \in \text{CFL}$

1. Schritt Modifikation des KA

- Baue K so um, dass
 1. $F = \{q_F\}$
 2. akzeptiert wird nur mit leerem Keller
 3. nur Push oder Pop (kein Push+Pop)
- Für 1. und 2. führe 3 neue Zustände q'_0, q_F, q_P ein

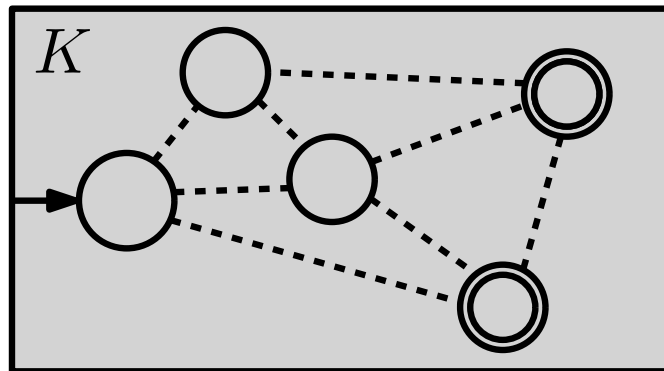
2. Teil des Beweises

Für jeden KA $K = (Q, \Sigma, \Gamma, \delta, q_0, F)$ gilt $L(K) \in \text{CFL}$

1. Schritt Modifikation des KA

- Baue K so um, dass
 1. $F = \{q_F\}$
 2. akzeptiert wird nur mit leerem Keller
 3. nur Push oder Pop (kein Push+Pop)
- Für 1. und 2. führe 3 neue Zustände q'_0, q_F, q_P ein

Schema



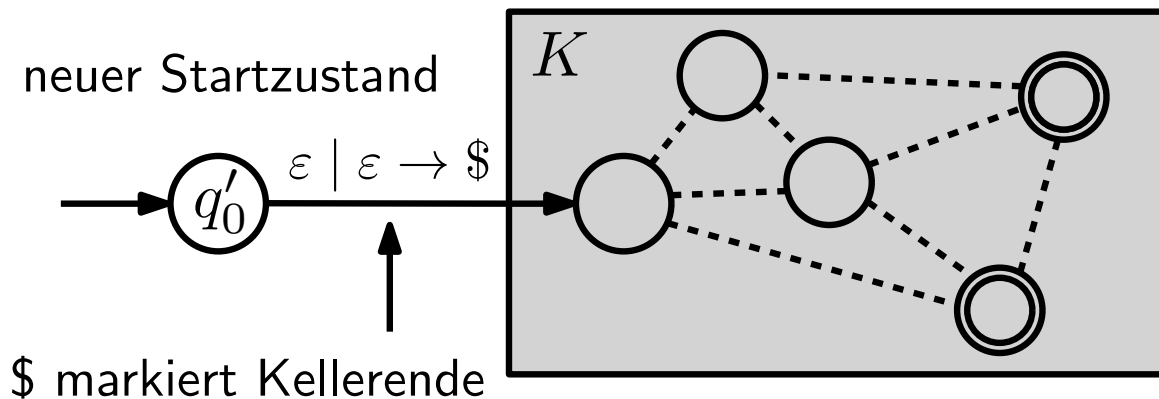
2. Teil des Beweises

Für jeden KA $K = (Q, \Sigma, \Gamma, \delta, q_0, F)$ gilt $L(K) \in \text{CFL}$

1. Schritt Modifikation des KA

- Baue K so um, dass
 1. $F = \{q_F\}$
 2. akzeptiert wird nur mit leerem Keller
 3. nur Push oder Pop (kein Push+Pop)
- Für 1. und 2. führe 3 neue Zustände q'_0, q_F, q_P ein

Schema



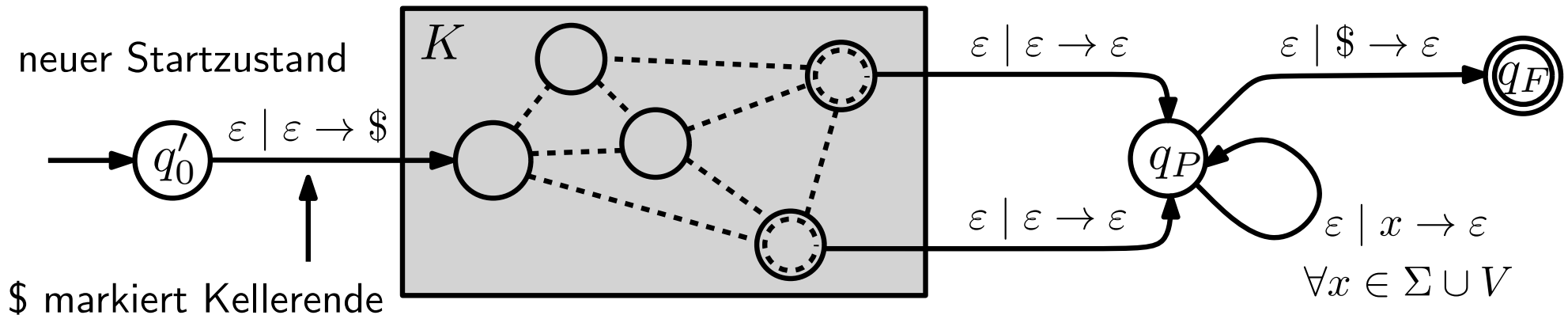
2. Teil des Beweises

Für jeden KA $K = (Q, \Sigma, \Gamma, \delta, q_0, F)$ gilt $L(K) \in \text{CFL}$

1. Schritt Modifikation des KA

- Baue K so um, dass
 1. $F = \{q_F\}$
 2. akzeptiert wird nur mit leerem Keller
 3. nur Push oder Pop (kein Push+Pop)

- Für 1. und 2. führe 3 neue Zustände q'_0, q_F, q_P ein
- Schema



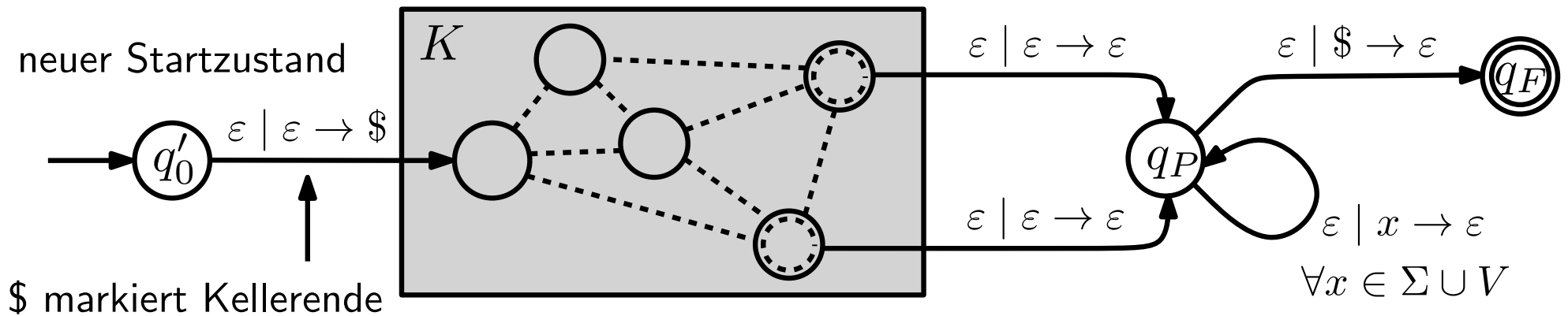
2. Teil des Beweises

Für jeden KA $K = (Q, \Sigma, \Gamma, \delta, q_0, F)$ gilt $L(K) \in \text{CFL}$

1. Schritt Modifikation des KA

- Baue K so um, dass
 1. $F = \{q_F\}$
 2. akzeptiert wird nur mit leerem Keller
 3. nur Push oder Pop (kein Push+Pop)
- Für 1. und 2. führe 3 neue Zustände q'_0, q_F, q_P ein

Schema

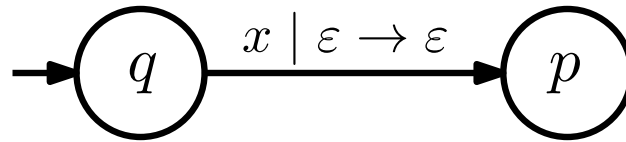


hier wird der Keller leergepumpt
nur bei leerem Keller wird akzeptiert

- Für 3. (jeder Übergang ist Push oder Pop)

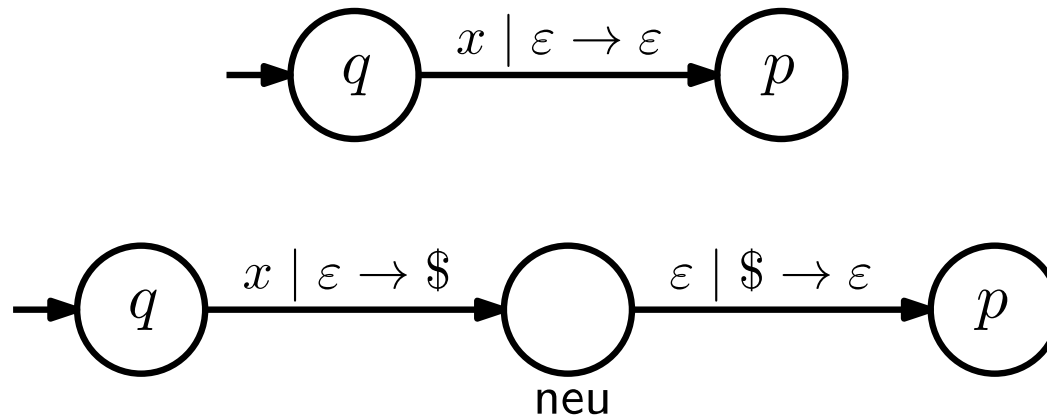
- Für 3. (jeder Übergang ist Push oder Pop)

Befehle die Keller unberührt lassen



- Für 3. (jeder Übergang ist Push oder Pop)

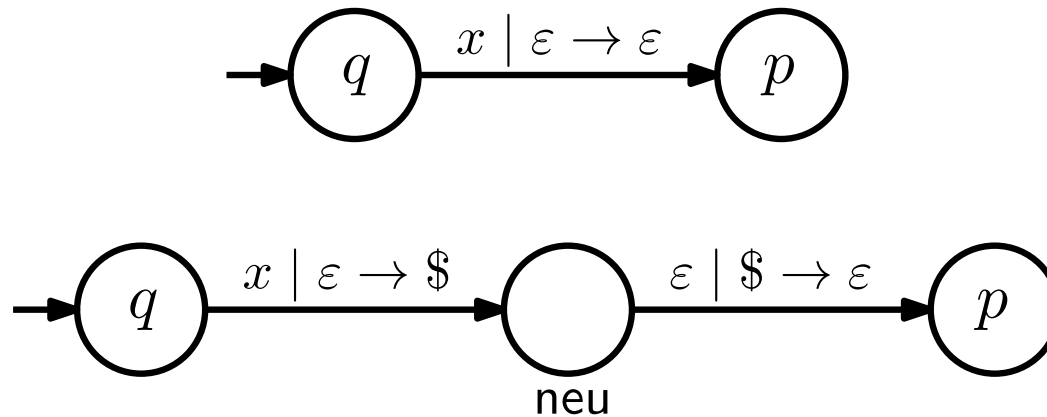
Befehle die Keller unberührt lassen



wird zu

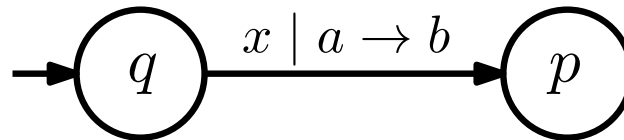
- Für 3. (jeder Übergang ist Push oder Pop)

Befehle die Keller unberührt lassen



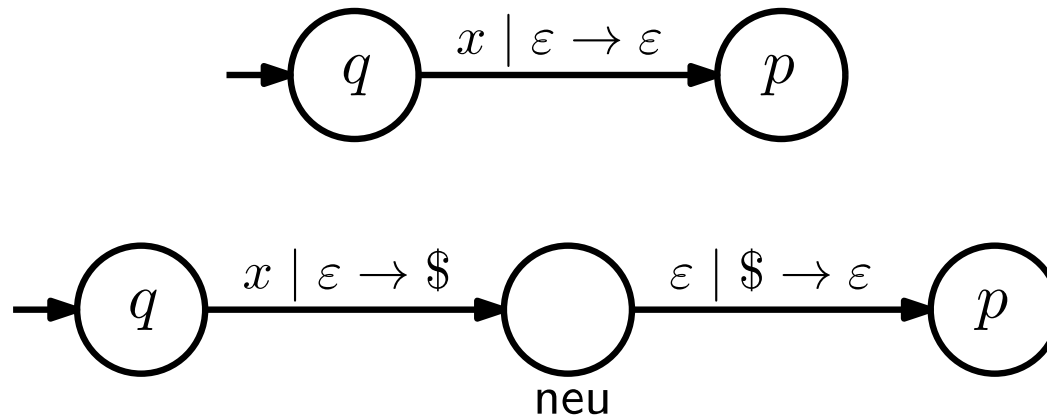
wird zu

Befehle Push+Pop (simultan)



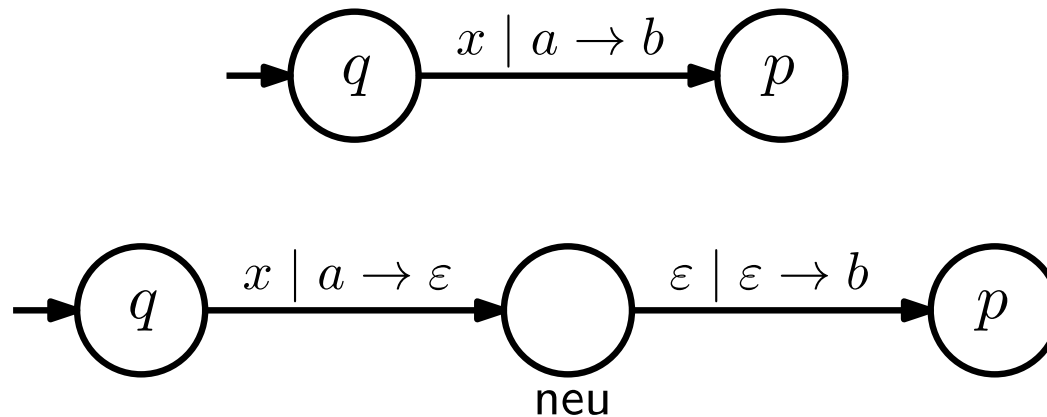
- Für 3. (jeder Übergang ist Push oder Pop)

Befehle die Keller unberührt lassen



wird zu

Befehle Push+Pop (simultan)



wird zu

Konstruktion Grammatik für $L(K)$

Konstruktion Grammatik für $L(K)$

- Variablen in G haben die Form A_{pq} , für jedes Zustandspaar p, q

Konstruktion Grammatik für $L(K)$


- Variablen in G haben die Form A_{pq} , für jedes Zustandspaar p, q
- Interpretation:

$\{w \mid A_{pq} \Rightarrow^* w\} =$ Wörter die Zustand p nach q mit leerem Keller überführen $(p \rightarrow_{\ell}^* q)$

Konstruktion Grammatik für $L(K)$


- Variablen in G haben die Form A_{pq} , für jedes Zustandspaar p, q
- Interpretation:

$\{w \mid A_{pq} \Rightarrow^* w\} =$ Wörter die Zustand p nach q mit leerem Keller überführen $(p \rightarrow_{\ell}^* q)$

 Keller am Anfang und am Ende leer

Konstruktion Grammatik für $L(K)$


- Variablen in G haben die Form A_{pq} , für jedes Zustandspaar p, q
- Interpretation:

$\{w \mid A_{pq} \Rightarrow^* w\} =$ Wörter die Zustand p nach q mit leerem Keller überführen $(p \rightarrow_{\ell}^* q)$
 Keller am Anfang und am Ende leer

- es gibt 2 Möglichkeiten für $p \rightarrow_{\ell}^* q$

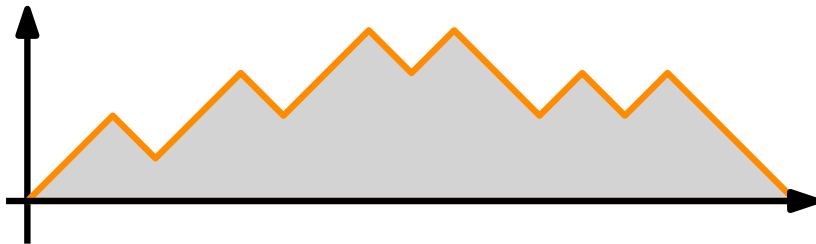
Konstruktion Grammatik für $L(K)$

- Variablen in G haben die Form A_{pq} , für jedes Zustandspaar p, q
- Interpretation:

$\{w \mid A_{pq} \Rightarrow^* w\} =$ Wörter die Zustand p nach q mit leerem Keller überführen $(p \rightarrow_{\ell}^* q)$
 Keller am Anfang und am Ende leer


- es gibt 2 Möglichkeiten für $p \rightarrow_{\ell}^* q$

1. Keller wird niemals leer



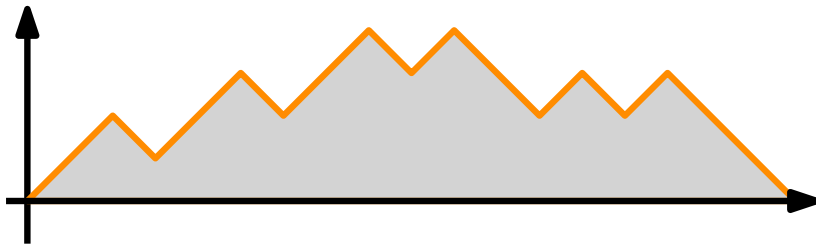
Konstruktion Grammatik für $L(K)$

- Variablen in G haben die Form A_{pq} , für jedes Zustandspaar p, q
- Interpretation:

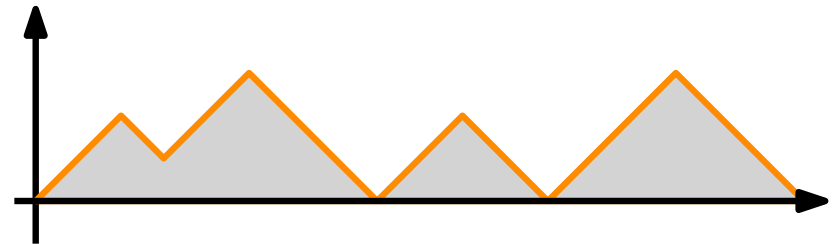
$\{w \mid A_{pq} \Rightarrow^* w\} =$ Wörter die Zustand p nach q mit leerem Keller überführen $(p \rightarrow_{\ell}^* q)$
 Keller am Anfang und am Ende leer

- es gibt 2 Möglichkeiten für $p \rightarrow_{\ell}^* q$

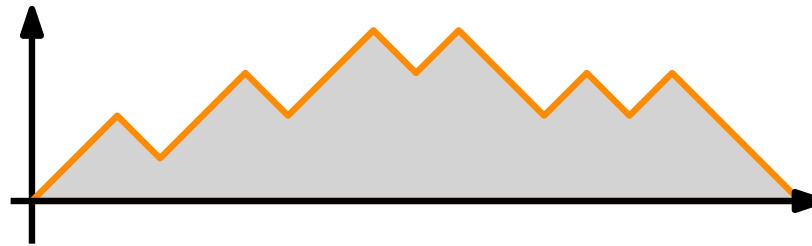
1. Keller wird niemals leer



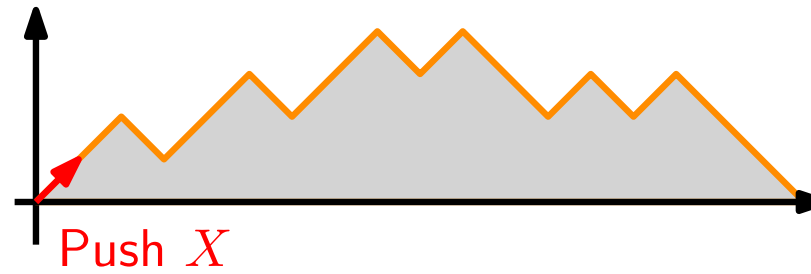
2. Keller wird leer



1. Keller wird niemals leer

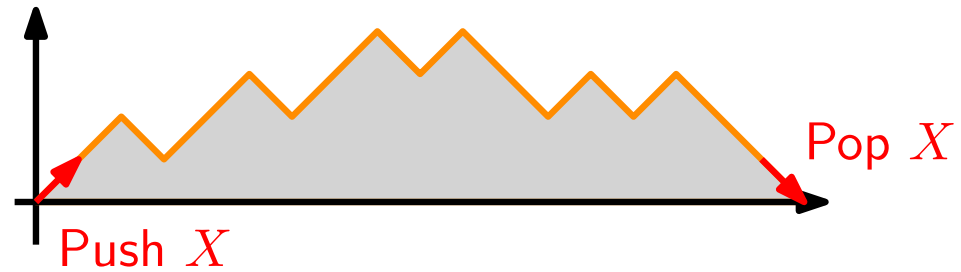


1. Keller wird niemals leer



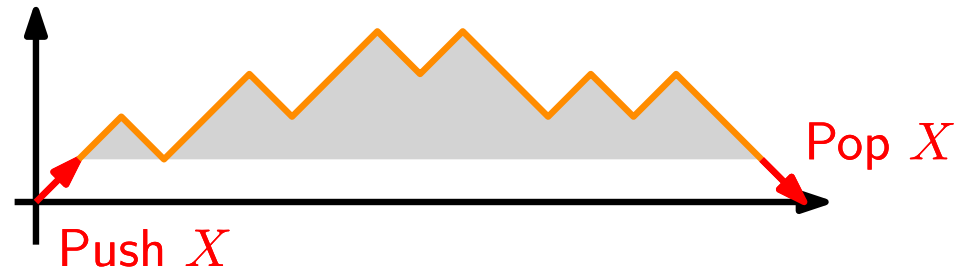
- erster Übergang war ein Push (z.B. Push X)

1. Keller wird niemals leer



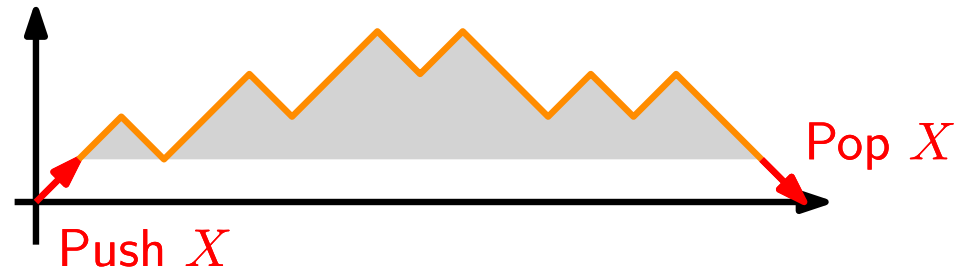
- erster Übergang war ein Push (z.B. Push X)
- letzter Übergang war ein Pop (konkret Pop X)

1. Keller wird niemals leer



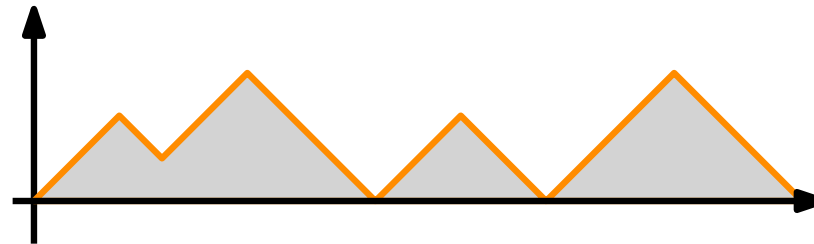
- erster Übergang war ein Push (z.B. Push X)
- letzter Übergang war ein Pop (konkret Pop X)
- wenn beim ersten Übergang a gelesen, Folgezustand r , beim letzten b gelesen, Vorzustand s , dann $A_{pq} \rightarrow aA_{rs}b$

1. Keller wird niemals leer

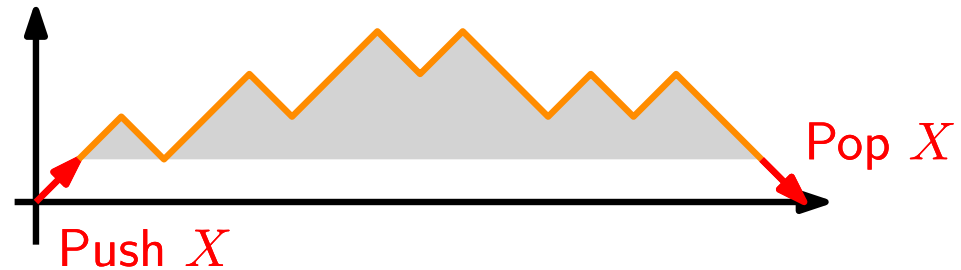


- erster Übergang war ein Push (z.B. Push X)
- letzter Übergang war ein Pop (konkret Pop X)
- wenn beim ersten Übergang a gelesen, Folgezustand r , beim letzten b gelesen, Vorzustand s , dann $A_{pq} \rightarrow aA_{rs}b$

2. Keller wird leer

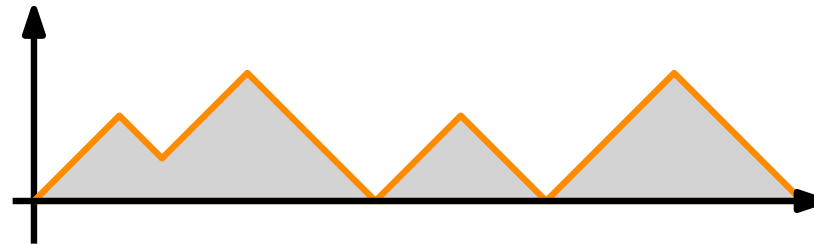


1. Keller wird niemals leer



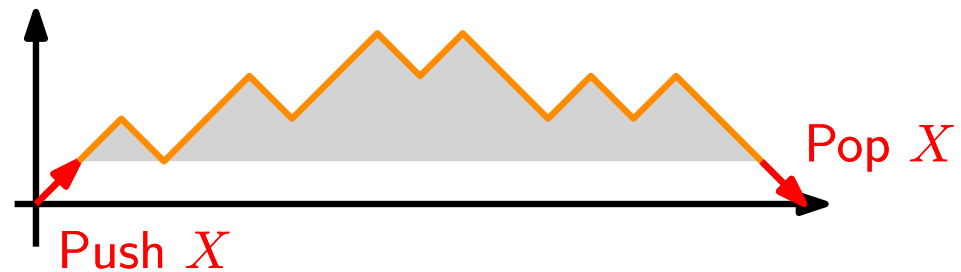
- erster Übergang war ein Push (z.B. Push X)
- letzter Übergang war ein Pop (konkret Pop X)
- wenn beim ersten Übergang a gelesen, Folgezustand r , beim letzten b gelesen, Vorzustand s , dann $A_{pq} \rightarrow aA_{rs}b$

2. Keller wird leer



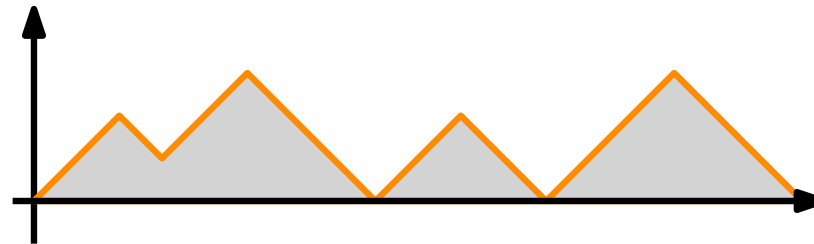
- im Zustand r wird der Keller leer

1. Keller wird niemals leer



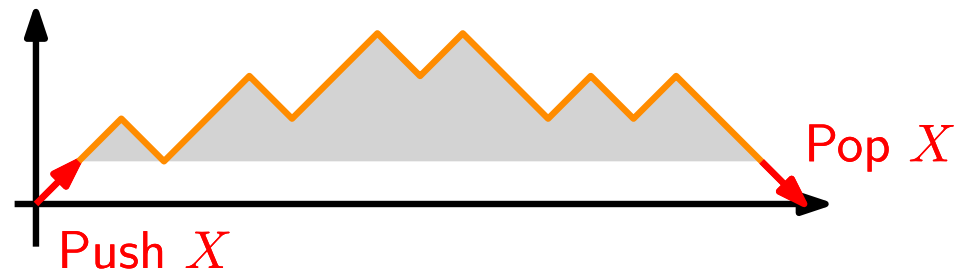
- erster Übergang war ein Push (z.B. Push X)
- letzter Übergang war ein Pop (konkret Pop X)
- wenn beim ersten Übergang a gelesen, Folgezustand r , beim letzten b gelesen, Vorzustand s , dann $A_{pq} \rightarrow aA_{rs}b$

2. Keller wird leer



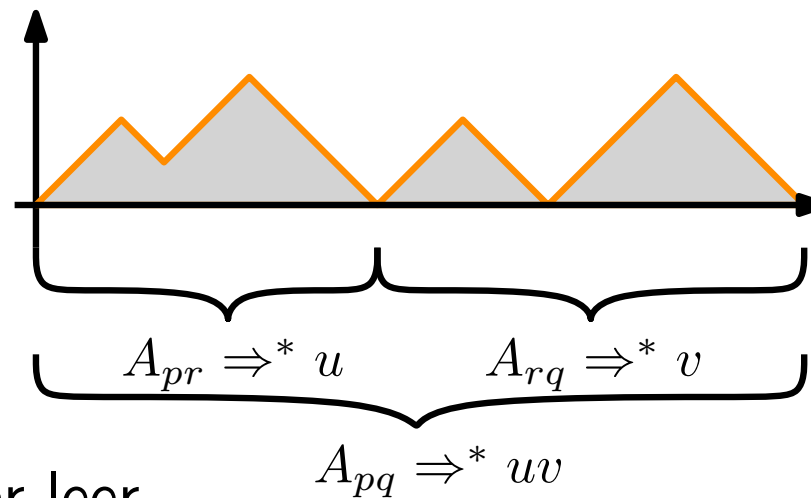
- im Zustand r wird der Keller leer
- dann gilt $A_{pq} \rightarrow A_{pr}A_{rq}$

1. Keller wird niemals leer



- erster Übergang war ein Push (z.B. Push X)
- letzter Übergang war ein Pop (konkret Pop X)
- wenn beim ersten Übergang a gelesen, Folgezustand r , beim letzten b gelesen, Vorzustand s , dann $A_{pq} \rightarrow aA_{rs}b$

2. Keller wird leer



- im Zustand r wird der Keller leer
- dann gilt $A_{pq} \rightarrow A_{pr}A_{rq}$