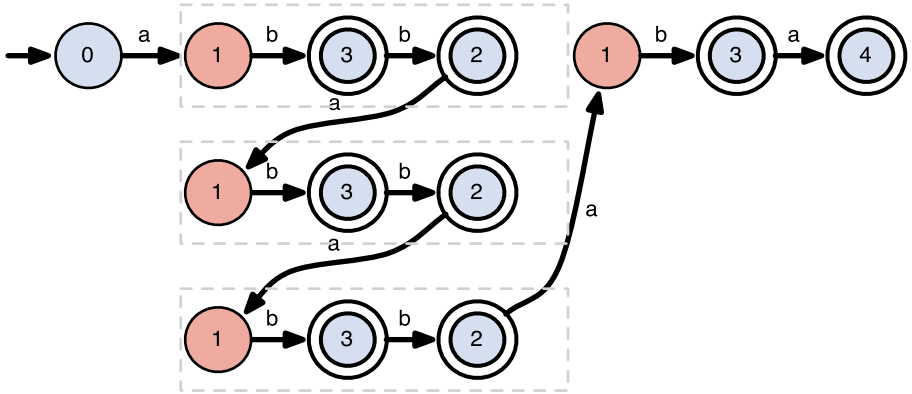
Berechenbarkeitstheorie

9. Vorlesung



Dr. Franziska Jahnke
Institut für Mathematische Logik und Grundlagenforschung
WWU Münster

Alle Folien unter Creative Commons Attribution-NonCommercial 3.0 Unported Lizenz.

CYK Algorithmus

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

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

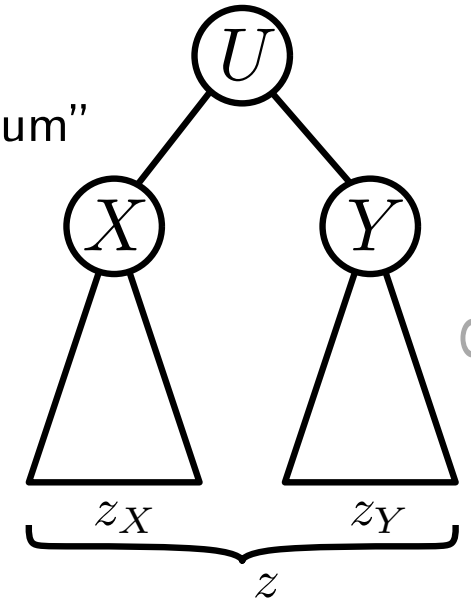
- $V_{ij} := \{U \in V \mid U \Rightarrow^* w[i,j]\}$
- **Teilprobleme des DP:** Bestimme die Mengen V_{ij}
- $w \in L \iff S \in V_{1,n}$, mit $|w| = n$
- Initialisierung: $V_{ii} = \{X \mid X \to w[i,i]\}$ für $i = 1, \ldots, n$
- Rekursiver Schritt: Bestimme V_{ij} mit Hilfe bereits bekannter Teilprobleme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

Finden einer Rekursion

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

  $U \Rightarrow XY \Rightarrow \cdots \Rightarrow z$

  2 Variablen da CNF



"Ableitungsbaum"

- 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 $w \neq \varepsilon$

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 ∈ V | X → 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
4          j = i + d;                                  aktueller Eintrag T[i,j]
5          for t = i to j − 1 do                       t ist Trennstelle
6              for all A → XY ∈ R do
7                  if X ∈ T[i,t] and Y ∈ T[t+1,j] then
                   T[i,j] = T[i,j] ∪ {A};
8              end
9          end
10     end
11 end
12 if S ∈ T[1,n] then return true;
13 else return false;
```

Bsp. 1

$$
\begin{array}{lcl}
S & \to & AB \mid CA \\
A & \to & AA \mid CB \mid \texttt{a} \\
B & \to & AC \mid \texttt{b} \\
C & \to & \texttt{c}
\end{array}
\qquad w = \texttt{cbaac}
$$

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

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

Bsp. 2

$$
\begin{array}{lcl}
S & \to & AB \mid CA \\
A & \to & AA \mid CB \mid \texttt{a} \\
B & \to & AC \mid \texttt{b} \\
C & \to & \texttt{c}
\end{array}
\qquad w = \texttt{aacaa}
$$

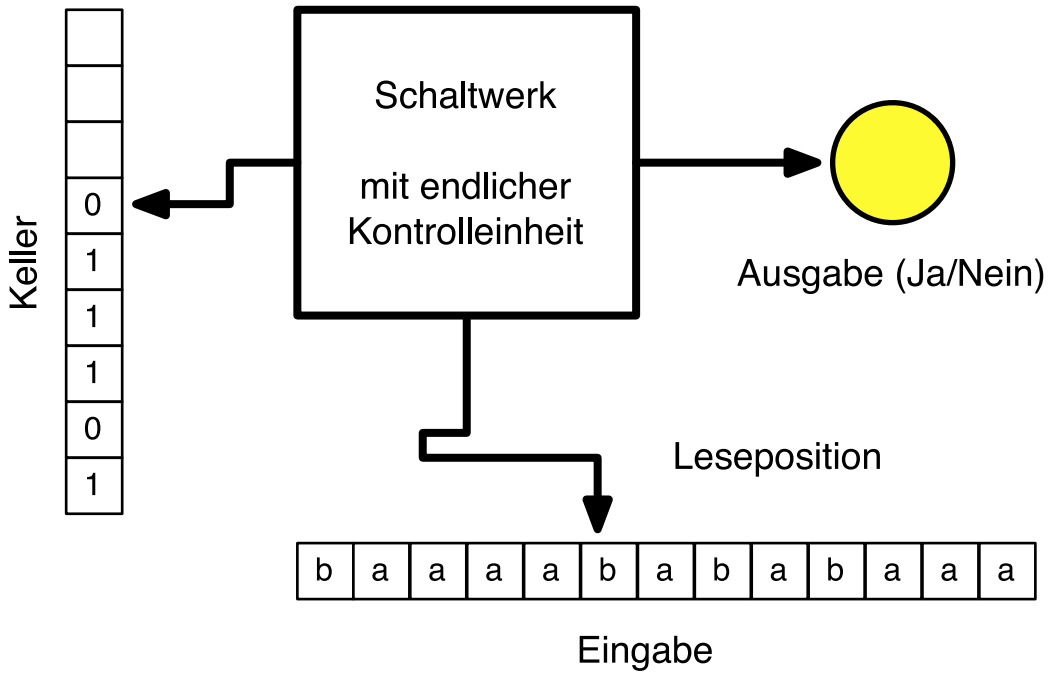
	1	2	3	4	5
5					A
4				A	A
3			C	S	S
2		A	B	–	–
1	A	A	S, B	–	–

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

Kellerautomaten

- **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 eingfügt
    - **Pop+Push**: Top-Symbol wird ausgetauscht
  - Nichtdeterminismus und ε-Übergänge bleiben erlaubt

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 $\delta\colon Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to \mathcal{P}(Q \times \Gamma_\varepsilon)$,
- $q_0 \in Q$, $F \subseteq Q$.

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

Interpretation Übergangsfunktion

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

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

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

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

Akzeptanzbegriff KA

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

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

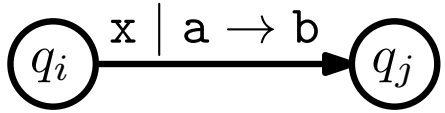
Bsp. Kellerautomat für $\{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\}$

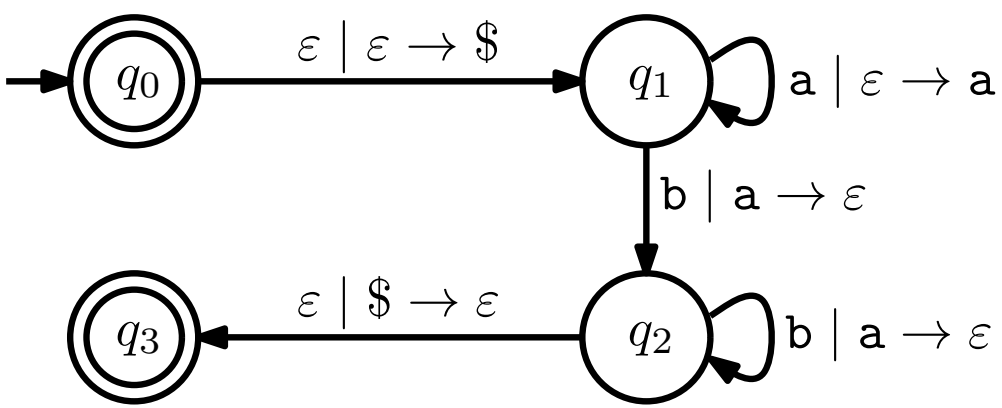
$Q = \{q_0, q_1, q_2, q_3\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}\}$ $\quad \Gamma = \{\$, \mathtt{a}\}$ $\quad F = \{q_0, q_3\}$

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

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





Satz 12

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

1. Teil des Beweis

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

Ansatz

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

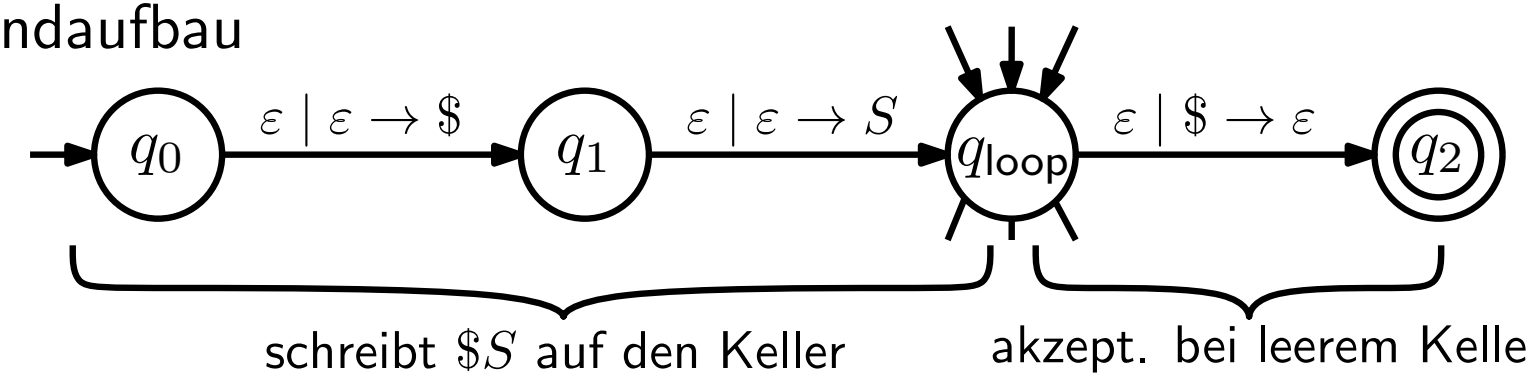
Hier Ableiten (Top-Symbol)

Muss mit w übereinstimmen, kann also schon geprüft werden

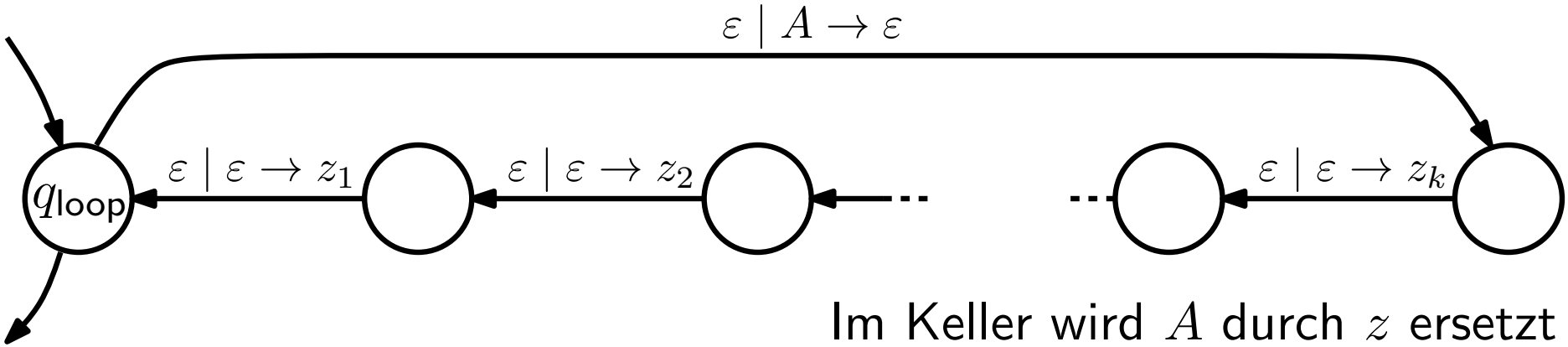
Konstruktion des KA

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

Grundaufbau

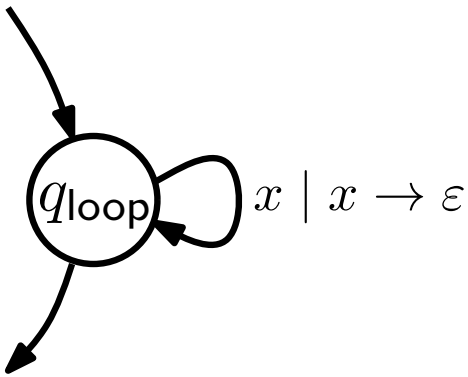


1. **Erweiterung:** Erlaubt Ableitungen des Top-Symbols
   - für jede Regel $A \to z = z_1 z_2 \cdots z_k$ $(z_i \in \Sigma \cup V)$ füge folgende Schleife hinzu



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

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

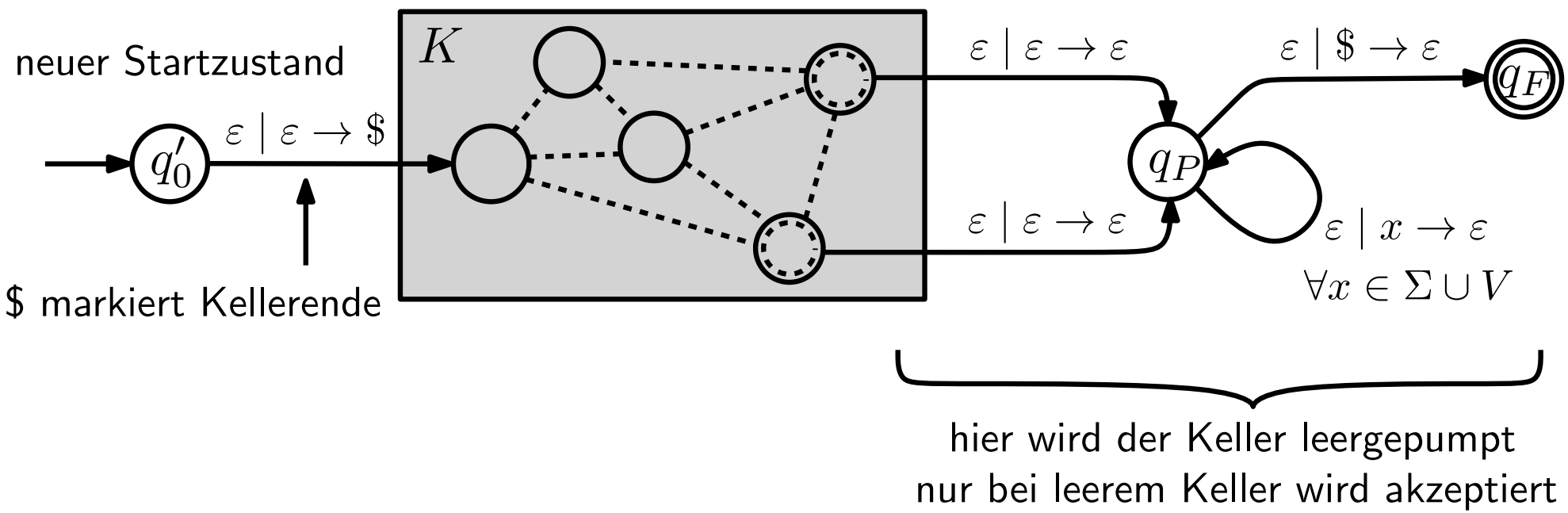
2. Teil des Beweises

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

1. Schritt Modifikation des KA

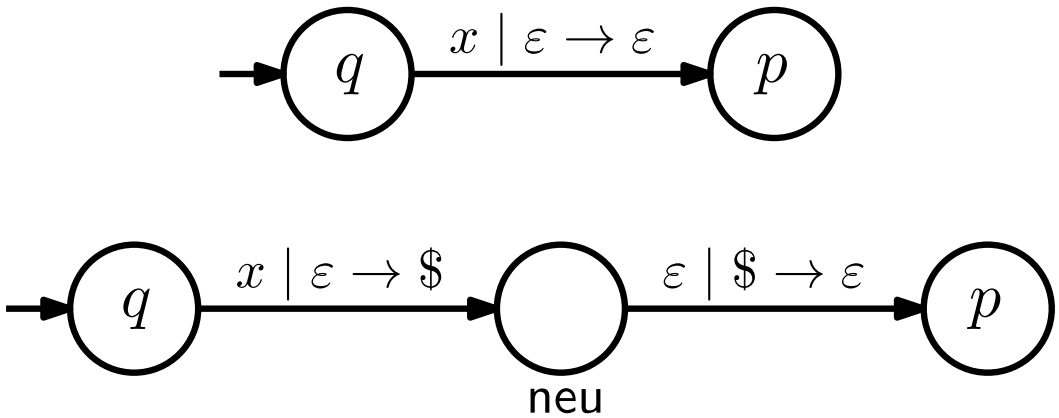
- Baue K so um, dass
  1. $F = \{q_F\}$
  2. akzeptiert wird nur mit leerem Keller
  3. nur Push oder Pop (kein Push+Pop)
- Für 1. und 2. führe 3 neue Zustände q_0', q_F, q_P ein

Schema

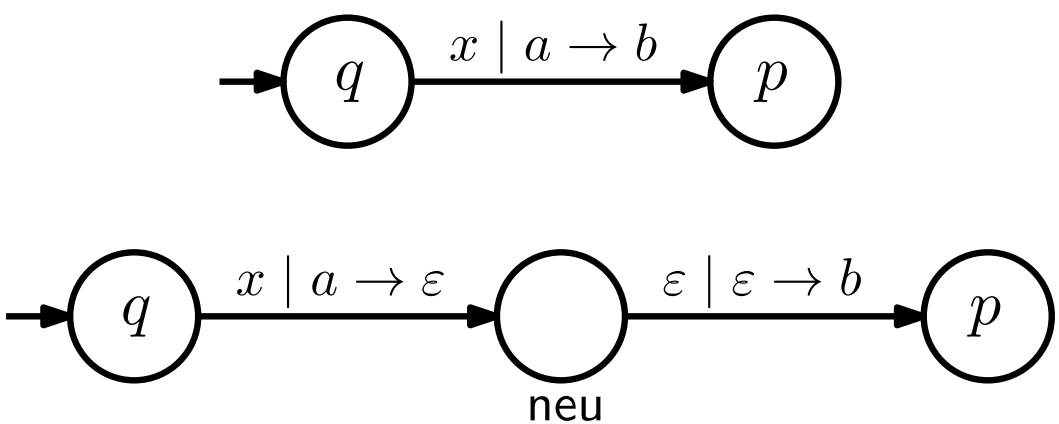


- Für 3. (jeder Übergang ist Push oder Pop)

Befehle die Keller unberührt lassen



Befehle Push+Pop (simultan)



Konstruktion Grammatik für $L(K)$

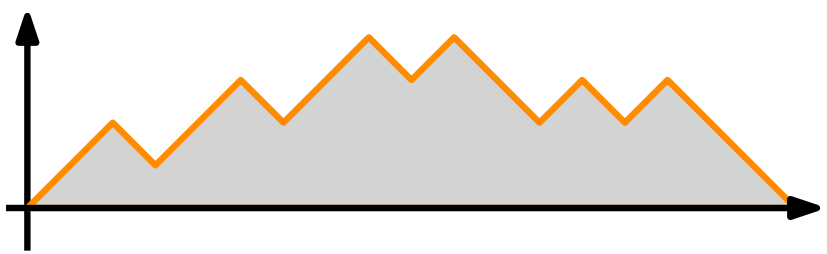
- Variablen in G haben die Form A_{pq}, für jedes Zustandspaar p, q
- Interpretation:

$$\{w \mid A_{pq} \Rightarrow^* w\} = \begin{array}{l}\text{Wörter die Zustand } p \text{ nach } q \\ \text{mit leerem Keller überführen}\end{array} \qquad (p \to_\ell^* q)$$

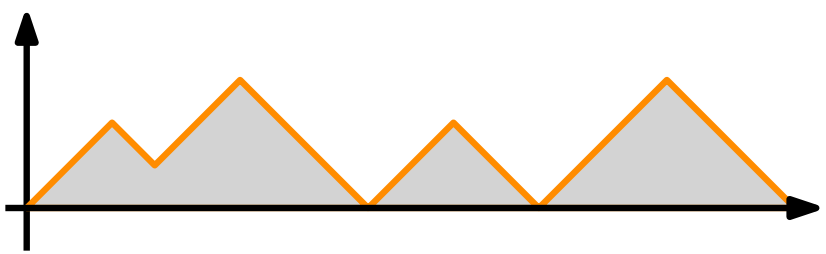
→ Keller am Anfang und am Ende leer

- es gibt 2 Möglichkeiten für $p \to_\ell^* q$

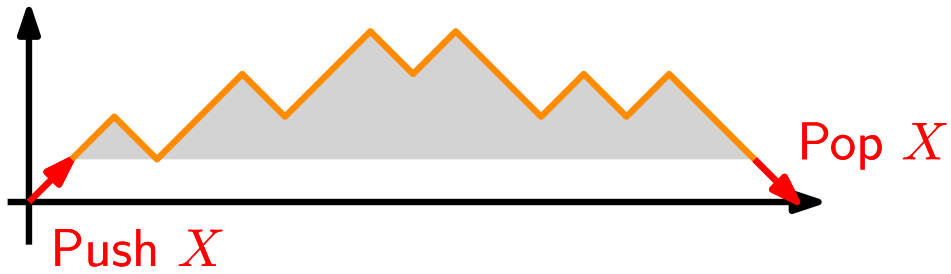
1. Keller wird niemals leer



2. Keller wird leer

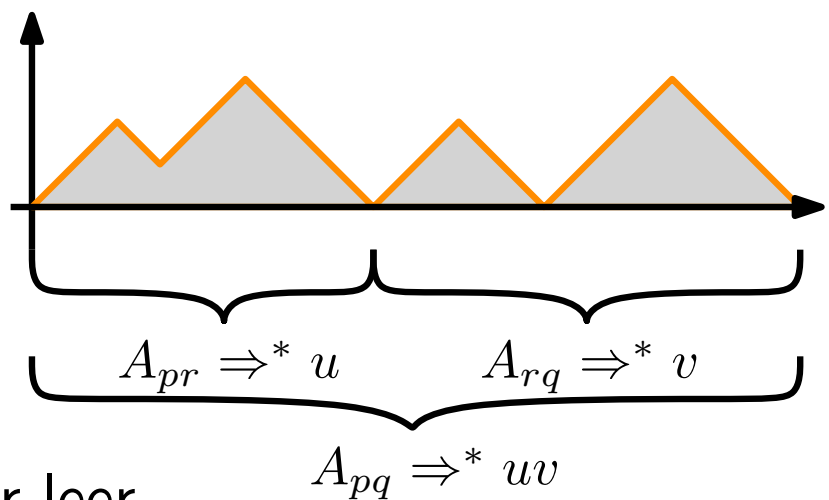


1. Keller wird niemals leer



- 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 $A_{pq} \to \mathtt{a} A_{rs} \mathtt{b}$

2. Keller wird leer



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