

# Tutorium Woche 3

Dominik Bruhn, Tutorium Nummer 9

11.11.2009

# Agenda

- 1 Definition
  - Vergleich regulär, kontextfrei
  - Kellerautomat
- 2 Pumping Lema
  - Definition
  - Aufgabe
- 3 Kontextfreie Sprachen
  - Definition
  - Aufgabe
  - CYK-Algorithmus
- 4 Ende

# Übersicht Sprache

## Reguläre Sprache

- Regeln:  
 $A \rightarrow aB$   
 $A \rightarrow a$
- Erkannt von endlichem Automat / DEA

## kontextfreie Sprache

- Regeln:  
 $A \rightarrow \text{irgendwas}$   
 $A \rightarrow \epsilon$
- Erkannt von nichtdet. Kellerautomat / KA

Jede **reguläre Sprache** ist auch eine **kontextfreie Sprache**. Dies gilt jedoch i.A. nicht umgekehrt.



# Kellerautomat

$\mathcal{K} = (Q, \Sigma, \Gamma, \delta, q_0, \mathcal{F})$  mit

- $Q$



# Kellerautomat

$\mathcal{K} = (Q, \Sigma, \Gamma, \delta, q_0, \mathcal{F})$  mit

- $Q$  Zustandsmenge
- $\Sigma$

# Kellerautomat

$\mathcal{K} = (Q, \Sigma, \Gamma, \delta, q_0, \mathcal{F})$  mit

- $Q$  Zustandsmenge
- $\Sigma$  Eingabealphabet
- $\Gamma$

# Kellerautomat

$\mathcal{K} = (Q, \Sigma, \Gamma, \delta, q_0, \mathcal{F})$  mit

- $Q$  Zustandsmenge
- $\Sigma$  Eingabealphabet
- $\Gamma$  Keller-Alphabet
- $\delta$

# Kellerautomat

$\mathcal{K} = (Q, \Sigma, \Gamma, \delta, q_0, \mathcal{F})$  mit

- $Q$  Zustandsmenge
- $\Sigma$  Eingabealphabet
- $\Gamma$  Keller-Alphabet
- $\delta$  Übergangsfunktion
- $q_0$

# Kellerautomat

$\mathcal{K} = (Q, \Sigma, \Gamma, \delta, q_0, \mathcal{F})$  mit

- $Q$  Zustandsmenge
- $\Sigma$  Eingabealphabet
- $\Gamma$  Keller-Alphabet
- $\delta$  Übergangsfunktion
- $q_0$  Startzustand
- $\mathcal{F}$

# Kellerautomat

$\mathcal{K} = (Q, \Sigma, \Gamma, \delta, q_0, \mathcal{F})$  mit

- $Q$  Zustandsmenge
- $\Sigma$  Eingabealphabet
- $\Gamma$  Keller-Alphabet
- $\delta$  Übergangsfunktion
- $q_0$  Startzustand
- $\mathcal{F}$  Menge der Endzustände

Anmerkungen:

- Keller (deutsch) = Stack (englisch)
- Per Definition ist der Keller zu Beginn leer
- Per Definition endet der Automat in jedem Endzustand (unabhängig vom Keller)

# Definition Pumping Lema

Sei  $L$  eine Sprache über einem Alphabet  $\Sigma$ .

## Definition (Pumping Lema)

$L$  regulär  $\Rightarrow \exists n \in \mathbb{N} : \forall \omega \in L, |\omega| \geq n : \exists \alpha, \beta, \gamma \in \Sigma^*, \omega = \alpha\beta\gamma :$   
 $\beta \neq \lambda \wedge |\alpha\beta| \leq n \wedge \forall i \in \mathbb{N}_0 : \alpha\beta^i\gamma \in L$

## Definition (Umkehrung des Pumping Lemmas)

$\forall n \in \mathbb{N} : \exists \omega \in L, |\omega| \geq n : \forall \alpha, \beta, \gamma \in \Sigma^*, \omega = \alpha\beta\gamma :$   
 $\beta = \lambda \vee |\alpha\beta| > n \vee \exists i \in \mathbb{N}_0 : \alpha\beta^i\gamma \notin L \Rightarrow L$  nicht regulär

# Pumping Lema Aufgabe

## Definition (Umkehrung des Pumping Lemmas)

$\forall n \in \mathbb{N} : \exists \omega \in L, |\omega| \geq n : \forall \alpha, \beta, \gamma \in \Sigma^*, \omega = \alpha\beta\gamma :$   
 $\beta = \lambda \vee |\alpha\beta| > n \vee \exists i \in \mathbb{N}_0 : \alpha\beta^i\gamma \notin L \Rightarrow L$  nicht regulär

## Aufgabe

Gegeben sei die Sprache

$\mathcal{L} = \{w \in \{a, b\}^* \mid w \text{ enthält gleich viele } a \text{ wie } b\}.$

Zeige mit Hilfe des Pumping Lemmas, dass  $L$  nicht regulär ist.

# Chomsky-Normalform

## Definition (Chomsky-Normalform)

Jede kontextfreie Sprache lässt sich in Chomsky-Normalform verwandeln:

- $A \rightarrow BC$
- $A \rightarrow a$
- $S \rightarrow \epsilon$  ( $S = \text{Startzeichen}$ )

# Aufgabe

## Gegeben

$\mathcal{G} = (\mathcal{T}, \mathcal{V}, S, \mathcal{P})$  mit

$\mathcal{T} := \{a, b, c, d\}$ ,  $\mathcal{V} := \{S, A, D, M\}$ ,  $\mathcal{P} := \{$

$S \rightarrow AMD \mid M$

$A \rightarrow AA \mid a$

$D \rightarrow DD \mid d$

$M \rightarrow bMc \mid \epsilon\}$

## Aufgabe

Wandele diese Grammatik in **Chomsky-Normalform** um

# Der CYK-Algorithmus

Der **Cocke-Younger-Kasami**-Algorithmus:

- Erwartet eine Grammatik in **Chomsky-Normalform** als Eingabe
- Überprüft ob sich ein Wort durch diese Grammatik erzeugen lässt
- Ist ein Beispiel für **Dynamische Programmierung**

# Aufgabe

## Gegeben

$\mathcal{G} = (\mathcal{T}, \mathcal{V}, S, \mathcal{P})$  mit  $\mathcal{T} := \{a, b, c, d\}$ ,  
 $\mathcal{V} := \{S, A, B, C, D, M, X, Y\}$ ,  $\mathcal{P} := \{$   
 $S \rightarrow AX \mid AD \mid BY \mid BC \mid \lambda$   
 $A \rightarrow AA \mid a$   
 $D \rightarrow DD \mid d$   
 $M \rightarrow BY \mid BC$   
 $X \rightarrow MD$   
 $Y \rightarrow MC$   
 $B \rightarrow b, C \rightarrow c\}$

## Aufgabe

Überprüfe ob  $aabbccdd$  durch  $\mathcal{G}$  erzeugt werden kann.

## Ende

# Fragen?

*Hätte ich, für jedes "Hätte ich",  
jedesmal nur 50 Center gekriegt  
Wär ich ein reicher Mann, der  
sichs leisten kann, auf den ganzen  
Scheiß hier zu scheißen, Mann!*  
Udo Lindenberg