

# FORMALE SYSTEME

## 16. Vorlesung: Kellerautomaten & CFGs

Markus Krötzsch  
Lehrstuhl Wissensbasierte Systeme

TU Dresden, 4. Dezember 2017

## Vorschau

## Ankündigung

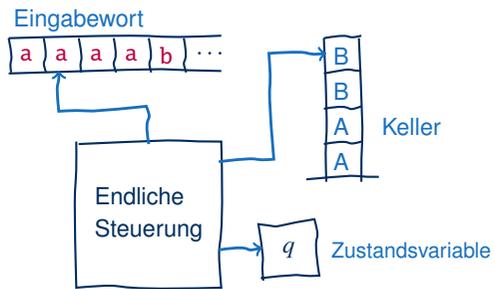
An Donnerstag, dem 14.12.2017, findet das erste Repetitorium statt

- Besprechung der Lösungen zu Zusatzaufgaben auf vorangegangenen Übungsblättern
- Aufgaben online verfügbar
- Am gewohnten Ort: DS4, HSZ/0003
- Präsentiert von Maximilian Marx

## Plan

- Grammatiken
- Reguläre Sprachen
  - DFA und NFA
  - Umformungen, Abschlusseigenschaften, Minimalisierung
  - Pumping Lemma
- Kontextfreie Sprachen
  - Wortproblem (CYK)
  - Pumping Lemma
  - Kellerautomaten
- Typ 1 und Typ 0
  - Turingmaschinen
  - (Un)Entscheidbarkeit
- Aussagenlogik
  - Syntax und Semantik
  - Algorithmen zum logischen Schließen
  - Die Komplexitätsklasse NP

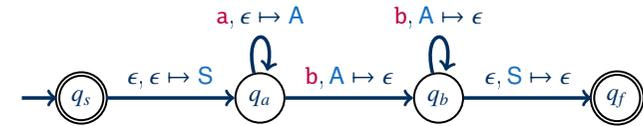
## Kellerautomat = NFA + Stapelspeicher



Eine mögliche **Konfiguration** des PDA während der Erkennung ist gegeben durch den Zustand  $q \in Q$ , den Inhalt des Kellers  $\gamma \in \Gamma^*$  und das noch zu lesende Restwort  $w \in \Sigma^*$ .

## Kellerautomaten (PDAs)

Beispiel eines PDA für  $\{a^i b^j \mid i \geq 0\}$ :



Ein **Kellerautomat** (international: „PDA“/„Pushdown Automaton“)  $\mathcal{M}$  ist ein Tupel  $\mathcal{M} = \langle Q, \Sigma, \Gamma, \delta, Q_0, F \rangle$  mit Zustandsmenge  $Q$ , Eingabealphabet  $\Sigma$ , Kellularphabet  $\Gamma$ , Startzuständen  $Q_0$ , Endzuständen  $F$  und Übergangsfunktion  $\delta$ :

$$Q \times \Sigma_\epsilon \times \Gamma_\epsilon \rightarrow 2^{Q \times \Gamma_\epsilon}$$

wobei  $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$  und  $\Gamma_\epsilon = \Gamma \cup \{\epsilon\}$ .

## Ausdrucksstärke von PDAs

## Ausdrucksstärke von PDAs

Man kann nun formal untersuchen, welche Sprachen durch PDAs akzeptiert werden können. Wir erhalten das erhoffte Resultat:

**Satz:** Eine Sprache ist genau dann kontextfrei wenn sie von einem PDA akzeptiert wird.

Der Beweis erfolgt in zwei Schritten:

- (1) Umwandlung Typ-2-Grammatik  $\rightsquigarrow$  PDA
- (2) Umwandlung PDA  $\rightsquigarrow$  Typ-2-Grammatik

# PDA $\rightsquigarrow$ Grammatik

Satz: Für jeden PDA  $\mathcal{P}$  gibt es eine kontextfreie Grammatik  $G_{\mathcal{P}}$ , so dass  $L(\mathcal{P}) = L(G_{\mathcal{P}})$ .

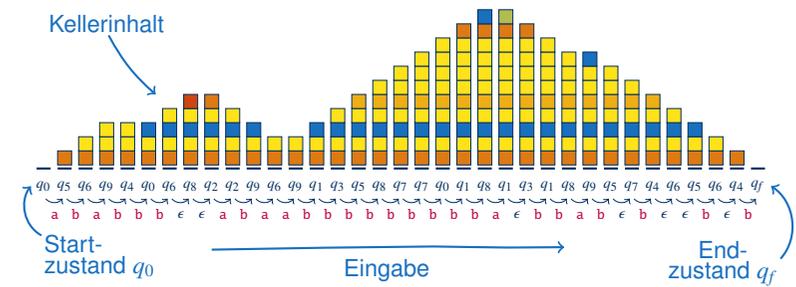
**Beweisidee:** Für jedes Paar von Zuständen  $q$  und  $r$  betrachten wir die Sprache  $L_{q,r}$ , die durch  $\mathcal{P}$  akzeptiert wird, wenn man mit leerem Keller in Zustand  $q$  beginnt und in Zustand  $r$  mit leerem Keller endet, d.h.

$$w \in L_{q,r} \text{ genau dann wenn } \langle q, w, \epsilon \rangle \vdash^* \langle r, \epsilon, \epsilon \rangle$$

- $L_{q,r}$  wird in der Grammatik durch eine Variable  $V_{q,r}$  dargestellt
  - Wir modifizieren  $\mathcal{P}$ , so dass
    - $\mathcal{P}$  je einen Startzustand  $q_0$  und einen Endzustand  $q_f$  hat
    - der Keller vor Erreichen von  $q_f$  geleert werden muss
- $\rightsquigarrow$  Die gesuchte Sprache  $L(\mathcal{P})$  ist genau  $L_{q_0, q_f}$

# Die Sprache $L_{q,r}$

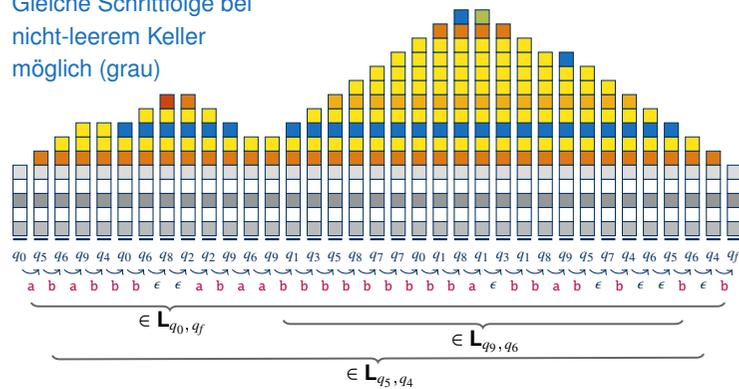
Schematische Darstellung der Konfigurationsfolge bei der Akzeptanz eines Wortes aus  $L_{q_0, q_f}$ :



# Die Sprache $L_{q,r}$ (2)

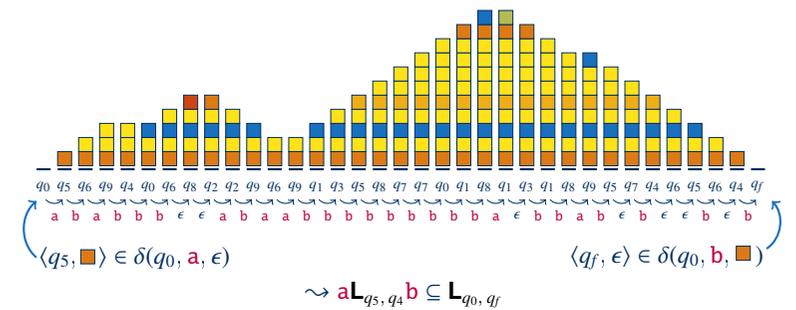
Schematische Darstellung der Konfigurationsfolge bei der Akzeptanz eines Wortes aus  $L_{q_0, q_f}$ :

Gleiche Schrittfolge bei nicht-leerem Keller möglich (grau)



# Die Sprache $L_{q,r}$ (3)

Schematische Darstellung der Konfigurationsfolge bei der Akzeptanz eines Wortes aus  $L_{q_0, q_f}$ :



## Eine Grammatik für $L_{q,r}$

Wir wollen für jedes  $L_{q,r}$  eine Variable  $V_{q,r}$  einführen, welche diese Sprache definiert

Aus dem gerade Gesehenen folgt:

- Wenn es ein Kellersymbol  $A$  gibt und
- wenn es ein Symbol  $a$  und einen Zustand  $s_1$  gibt, so dass  $\langle s_1, A \rangle \in \delta(q, a, \epsilon)$ , und
- wenn es ein Symbol  $b$  und einen Zustand  $s_2$  gibt, so dass  $\langle r, \epsilon \rangle \in \delta(s_2, b, A)$
- dann sollte die Grammatik die Regel

$$V_{q,r} \rightarrow aV_{s_1,s_2}b$$

enthalten.

## Eine Grammatik für $L_{q,r}$ (2)

Wir wollen für jedes  $L_{q,r}$  eine Variable  $V_{q,r}$  einführen, welche diese Sprache definiert

Aus dem gerade Gesehenen folgt:

- Für jeden Zustand  $s$
- enthält die Grammatik die Regel

$$V_{q,r} \rightarrow V_{q,s}V_{s,r}$$

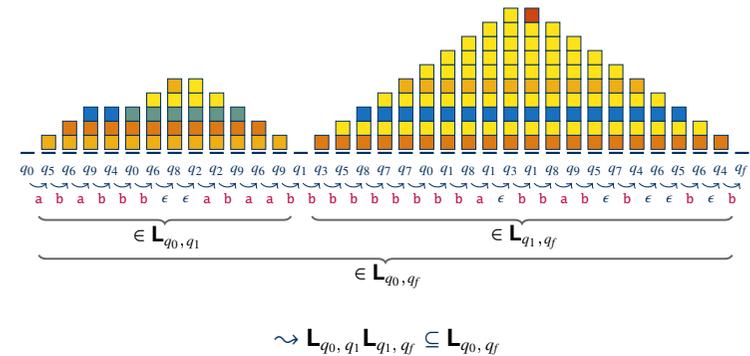
Außerdem benötigen wir noch die folgenden Regeln, damit die Ableitung irgendwann abgeschlossen werden kann:

- Für jeden Zustand  $s$
- enthält die Grammatik die Regel

$$V_{s,s} \rightarrow \epsilon.$$

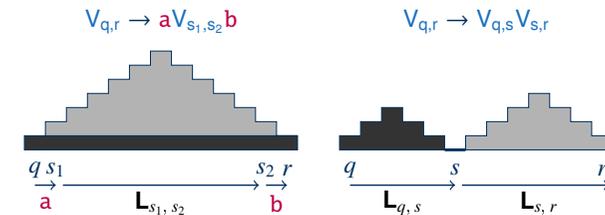
## Die Sprache $L_{q,r}$ (4)

Es gibt einen zweiten relevanten Fall zur Ableitung von  $L_{q_0, q_f}$ :



## Zwischenbilanz

Bisher haben wir zwei Arten von Konfigurationsfolgen bzw. Grammatikregeln eingeführt:



Man könnte auf diese beiden Arten einen beliebigen Lauf in immer kleinere Schritte zu zerlegen und somit das erkannte Wort generieren. Den Abschluss bilden die Regeln  $V_{s,s} \rightarrow \epsilon$ .

Funktioniert das im Prinzip für alle Läufe von  $\mathcal{P}$ ?

**Nein:** Damit kann man Läufe generieren, bei denen sich die Höhe des Kellers in jedem Schritt ändert. Bleibt der Keller über mehrere Schritte gleich hoch, dann hilft keine der Zerlegungen weiter.

## Abschluss des Beweises

**Problem:** Unsere Zerlegung funktioniert nicht, wenn der Keller über mehrere Schritte die gleiche Höhe behält.

**Lösung:** Wir können PDAs so abwandeln, dass sie in jedem Schritt pop oder push, aber nie beides ausführen.

### Skizze:

- Übergänge mit pop und push zerlegt man mithilfe eines Zwischenzustands in einen pop und einen push
- Übergänge ohne pop oder push, stellt man dar, indem man zunächst ein Hilfsszeichen pusht und dieses gleich darauf wieder popt

Mit diesen Änderungen gilt tatsächlich:

**Fakt:**  $V_{q,r}$  erzeugt ein Wort  $w$  genau dann wenn  $\mathcal{P}$  von  $q$  mit leerem Keller zu  $r$  mit leerem Keller gelangen kann.

(ausführlicher Beweis siehe Sipser, „Introduction to the Theory of Computation“ (Int. Ed.), Abschnitt 2.2)

## Der Beweis ist komplett

**Zusammenfassung:** Der PDA  $\mathcal{P}$  wird so abgewandelt, dass

- $\mathcal{P}$  je einen Startzustand  $q_0$  und einen Endzustand  $q_f$  hat
- der Keller vor Erreichen von  $q_f$  geleert werden muss
- $\mathcal{P}$  in jedem Schritt pop oder push ausführt, aber nie beides

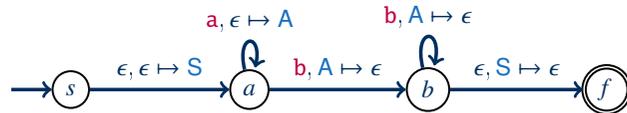
Die Grammatik  $G(\mathcal{P})$  besteht aus den folgenden Regeln:

- (1) Für alle  $q, r, s_1, s_2 \in Q, a, b \in \Sigma_\epsilon$  und  $A \in \Gamma$ , für die es Übergänge  $\langle s_1, A \rangle \in \delta(q, a, \epsilon)$  und  $\langle r, \epsilon \rangle \in \delta(s_2, b, A)$  gibt, die Regel  $V_{q,r} \rightarrow aV_{s_1,s_2}b$
- (2) Für alle Zustände  $q, r, s \in Q$ , die Regel  $V_{q,r} \rightarrow V_{q,s}V_{s,r}$
- (3) Für alle Zustände  $s \in Q$ , die Regel  $V_{s,s} \rightarrow \epsilon$

Dann ist  $L(\mathcal{P}) = L_{q_0, q_f}$ . Die Grammatik  $G(\mathcal{P})$  erzeugt diese Sprache, wenn wir  $V_{q_0, q_f}$  als Startsymbol wählen.  $\square$

## Beispiel

PDA für  $\{a^i b^i \mid i \geq 1\}$ :



Dieser PDA erfüllt die Anforderungen aus dem Beweis.

Als Grammatik ergibt sich:

$$\begin{aligned}
 V_{s,f} &\rightarrow \epsilon V_{a,b} \epsilon \\
 V_{a,b} &\rightarrow a V_{a,b} b \mid a V_{a,a} b \\
 V_{a,a} &\rightarrow \epsilon \quad V_{b,b} \rightarrow \epsilon \quad V_{s,s} \rightarrow \epsilon \quad V_{f,f} \rightarrow \epsilon \\
 V_{a,a} &\rightarrow V_{a,a} V_{a,a} \mid V_{a,b} V_{b,a} \mid V_{a,s} V_{s,a} \mid V_{a,f} V_{f,a} \quad V_{a,b} \rightarrow \dots \quad \dots
 \end{aligned}$$

(nicht alle dieser Regeln werden wirklich benötigt)

## Lehrevaluation

15min zum Ausfüllen der Fragebögen

# Deterministische Kellerautomaten

## Deterministische Kellerautomaten

Ein **deterministischer Kellerautomat** (international: „DPDA“)  $\mathcal{M}$  ist ein Tupel  $\mathcal{M} = \langle Q, \Sigma, \Gamma, \delta, q_0, F \rangle$  mit den folgenden Bestandteilen:

- $Q$ : endliche Menge von **Zuständen**
- $\Sigma$ : **Eingabealphabet**
- $\Gamma$ : **Kelleralphabet**
- $\delta$ : **Übergangsfunktion**, eine partielle Funktion

$$Q \times \Sigma_\epsilon \times \Gamma_\epsilon \rightarrow Q \times \Gamma_\epsilon,$$

so dass für alle  $q \in Q$ ,  $a \in \Sigma$  und  $A \in \Gamma$  jeweils nur eines der folgenden definiert ist:

$$\delta(q, a, A) \quad \delta(q, a, \epsilon) \quad \delta(q, \epsilon, A) \quad \delta(q, \epsilon, \epsilon)$$

- $q_0$ : ein **Startzustand**  $q_0 \in Q$
- $F$ : Menge von **Endzuständen**  $F \subseteq Q$

# Deterministische Kellerautomaten?

Es gibt verschiedene Quellen für Nichtdeterminismus bei PDAs:

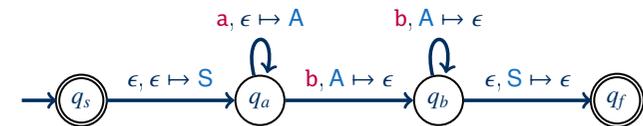
- Übergangsfunktion liefert eine Menge möglicher Übergänge
- Mehrere mögliche Startzustände
- $\epsilon$ -Übergänge in Eingabe und Keller

Beispiel: Angenommen es gilt  $\delta(q, \epsilon, A) = \{ \langle p, B \rangle \}$  und  $\delta(q, a, \epsilon) = \{ \langle r, C \rangle \}$ . Dann gibt es mehrere mögliche Übergänge, wenn in Zustand  $q$  das Symbol  $a$  gelesen wird und  $A$  auf dem Keller liegt. Und das obwohl die Übergangsfunktion nur ein-elementige Mengen liefert.

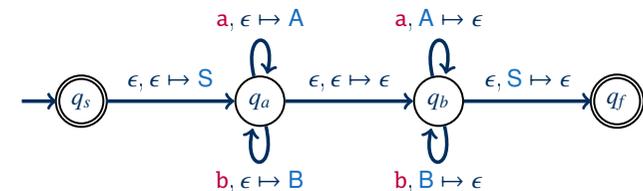
Deterministische Kellerautomaten müssen eingeschränkt werden, ohne  $\epsilon$ -Übergänge ganz zu verbieten (ganz ohne  $\epsilon$ -Übergänge sind PDAs zu schwach)

## Beispiele

Dieser Automat ist ein DPDA:



Dieser Automat ist **kein** DPDA:



## Deterministische kontextfreie Sprachen

DPDAs führen zu einer eigenen Sprachklasse:

Eine Sprache ist **deterministisch kontextfrei** wenn sie durch einen deterministischen Kellerautomaten akzeptiert wird.

Offensichtlich ist jede deterministisch kontextfreie Sprache auch kontextfrei (DPDAs können als PDAs gesehen werden).

Im Gegensatz zur Situation bei DFAs und NFAs gilt die Umkehrung dieser Aussage nicht:

**Satz:** Die deterministisch kontextfreien Sprachen bilden eine echte Untermenge der kontextfreien Sprachen.  
Es gibt also kontextfreie Sprachen, die nicht deterministisch sind.

## Nichtdeterministische kontextfreie Sprachen

Eine kontextfreie Sprache, deren Komplement nicht kontextfrei ist, kann demnach nicht deterministisch kontextfrei sein.

Beispiel: In Vorlesung 14 haben wir die Sprachen  $L_1 = \{a^i b^j c^k \mid i \geq 0, k \geq 0\}$  und  $L_2 = \{a^i b^k c^k \mid i \geq 0, k \geq 0\}$  betrachtet und erkannt, dass die Sprache

$$\overline{L_1} \cup \overline{L_2} = \{a^i b^j c^k \mid i \neq j \text{ oder } j \neq k\}$$

kontextfrei ist, aber ihr Komplement nicht. Diese Sprache ist demnach nicht deterministisch kontextfrei.

Das beweist im wesentlichen die Behauptung, dass die deterministisch kontextfreie Sprachen eine echte Untermenge der kontextfreien sind.

Es gibt noch viele andere nichtdeterministische Sprachen, z.B.:

Beispiel: Die Sprache  $\{ww^r \mid w \in \{a, b\}^*\}$  der Palindrome gerader Länge (ohne Markierung in der Mitte) ist kontextfrei aber nicht deterministisch.  
(Ohne Beweis; siehe z.B. Skript Franz Baader)

## Abschluss unter Komplement

Der Unterschied zu allgemeinen kontextfreien Sprachen zeigt sich gut am folgenden Ergebnis:

**Satz:** Die Klasse der deterministisch kontextfreien Sprachen ist unter Komplement abgeschlossen.

**Beweisidee:** Wie bei DFAs kann man auch bei DPDAs die akzeptierenden und nichtakzeptierenden Zustände vertauschen.

Mehrere technische Komplikationen machen den Beweis etwas aufwändiger:

- Ein DPDA kann ein Wort auf zwei Arten nicht akzeptieren: (1) nach dem Lesen der Eingabe ist der Automat nicht in einem Endzustand; (2) die Eingabe wird nie ganz gelesen, z.B. weil der Automat in eine Endlosschleife von  $\epsilon$ -Übergängen gerät. Man muss Grund (2) ausschalten.
- Ein Lauf könnte mit einer Folge von  $\epsilon$ -Übergängen enden, die akzeptierende und nichtakzeptierende Zustände enthält. In diesem Fall führt Austausch der Endzustände nicht zum gewünschten Ergebnis. Man muss sicherstellen, dass grundsätzlich (vor und nach der Komplementierung) nur der erste Zustand in einer Folge von  $\epsilon$ -Übergängen akzeptieren darf.

(vollständiger Beweis siehe Sipser, Abschnitt 2.4)

□

## Zusammenfassung und Ausblick

PDAs erkennen **genau die kontextfreien Sprachen**:

- $\text{PDA} \rightsquigarrow \text{Typ-2-Grammatik}$  durch Simulation von Linksableitungen
- $\text{Typ-2-Grammatik} \rightsquigarrow \text{PDA}$  durch rekursive Erzeugung der Sprachen  $L_{q,r}$

**Deterministische Kellerautomaten** (DPDAs) erkennen nur eine echte Untermenge der kontextfreien Sprachen.

Offene Fragen:

- Wozu sind deterministische kontextfreie Sprachen gut?
- Welche Probleme auf CFGs kann man lösen?
- Was gibt es zu Typ 1 und Typ 0 zu sagen?