

FORMALE SYSTEME

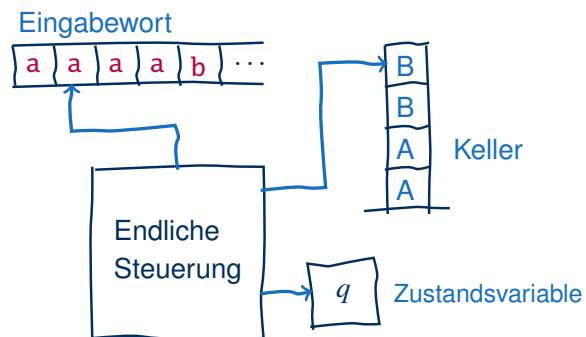
16. Vorlesung: Kellerautomaten & CFGs

Markus Krötzsch

Lehrstuhl Wissensbasierte Systeme

TU Dresden, 5. Dezember 2016

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 $w \in \Sigma^*$.

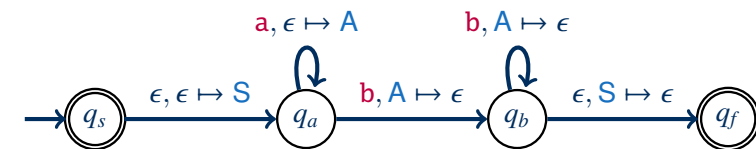
Ankündigung

An Donnerstag, dem 8.12.2016, findet das erste Repetitorium statt

- Besprechung der Lösungen zu *-Aufgaben auf vorangegangenen Übungsblättern
- Aufgaben online verfügbar
- Am gewohnten Ort: DS4, HSZ/0003
- Präsentiert von Francesco Kriegel

Kellerautomaten (PDAs)

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



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 Σ , Kelleralphabet Γ , Startzuständen Q_0 , Endzuständen F und Übergangsfunktion δ :

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

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

PDA: Variationen

Es gibt auch hier viele äquivalente Definitionen

- Ein Startzustand reicht auch aus (Übung: warum?)
- Alternative Akzeptanzbedingung: ersetze „PDA ist in Endzustand“ durch „PDA hat leeren Keller“ (keine Endzustände nötig, dafür manchmal aufwendigere Kodierung)
- Die Übergangsfunktion δ kann auch als Relation $\Delta \subseteq (Q \times \Sigma_\epsilon \times \Gamma_\epsilon) \times (Q \times \Gamma_\epsilon)$ beschrieben werden (wie bei NFAs)
- Das Startsymbol des Kellers kann Teil der Definition sein
- Man kann erlauben, dass in einem Schritt mehrere Symbole auf den Keller geschrieben werden.

Dann kann man auch verlangen, dass in jedem Schritt ein Symbol entnommen wird (pop), da man es bei Bedarf zurücklegen kann

Jede dieser Varianten führt zur gleichen Ausdrucksstärke

Ausdrucksstärke von PDAs

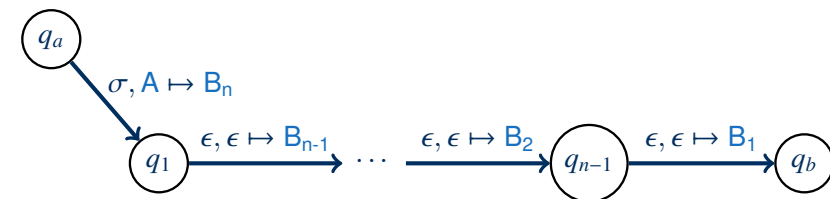
Wörter pushen

Wir können Übergänge der folgenden Form erlauben:



„Wenn in Zustand q_a das Zeichen (oder leere Wort) σ gelesen wird und wenn A oben vom Keller geholt werden kann (pop), dann wechsele in Zustand q_b und lege die Zeichen $B_1 \dots B_n$ auf den Keller (wobei B_1 ganz oben zu liegen kommt).“

Dies kann mithilfe neuer Zustände kodiert werden:



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

Grammatik \rightsquigarrow PDA

Wir wollen zunächst eine Richtung zeigen:

Satz: Für jede kontextfreie Grammatik G gibt es einen PDA \mathcal{P}_G , so dass $L(G) = L(\mathcal{P}_G)$.

Beweisidee:

- Der PDA simuliert eine Ableitung in der Grammatik
- Der Keller speichert das Zwischenergebnis der Ableitung (Terminale und Nichtterminale), linke Zeichen zuoberst
- Der PDA arbeitet das Zwischenergebnis von links ab:
 - Terminale werden mit Eingabe verglichen und entfernt
 - Nichtterminale werden durch Grammatikregel ersetzt
- Die Ableitung gelingt, wenn die gesamte Eingabe gelesen werden konnte und der Keller leer ist

Beispiel

Die Grammatik $S \rightarrow aSa \mid bSb \mid \epsilon$ ergibt den folgenden PDA:

$\epsilon, S \mapsto aSa$
 $\epsilon, S \mapsto bSb$
 $\epsilon, S \mapsto \epsilon$
 $a, a \mapsto \epsilon$
 $b, b \mapsto \epsilon$



Mögliche Konfigurationsfolge (Wörter werden zur Vereinfachung in einem Schritt auf Stack geschrieben):

$\langle q_s, abba, \epsilon \rangle \vdash \langle q_l, abba, S\# \rangle \vdash \langle q_l, abba, aSa\# \rangle \vdash \langle q_l, bba, Sa\# \rangle \vdash$
 $\langle q_l, bba, bSba\# \rangle \vdash \langle q_l, ba, Sba\# \rangle \vdash \langle q_l, ba, ba\# \rangle \vdash \langle q_l, a, a\# \rangle \vdash$
 $\langle q_l, \epsilon, \# \rangle \vdash \langle q_f, \epsilon, \epsilon \rangle$

(akzeptierender Lauf)

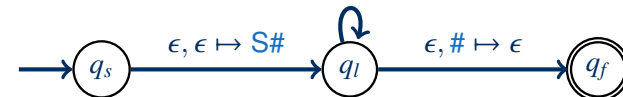
Grammatik \rightsquigarrow PDA

Wir wollen zunächst eine Richtung zeigen:

Satz: Für jede kontextfreie Grammatik G gibt es einen PDA \mathcal{P}_G , so dass $L(G) = L(\mathcal{P}_G)$.

Beweis: Sei $G = \langle V, \Sigma, P, S \rangle$. Wir definieren einen erweiterten PDA:

$\epsilon, A \mapsto w$ für jede Regel $A \rightarrow w$
 $a, a \mapsto \epsilon$ für jedes Symbol $a \in \Sigma$



Das heißt, wir haben:

$$Q = \{q_s, q_l, q_f\} \quad Q_0 = \{q_s\} \quad F = \{q_f\} \quad \Gamma = \Sigma \cup V \cup \{\#\}$$

Durch Entfernen der Wort-push-Operationen wie zuvor gezeigt entsteht der gesuchte PDA \mathcal{P}_G . □

PDA \rightsquigarrow Grammatik

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

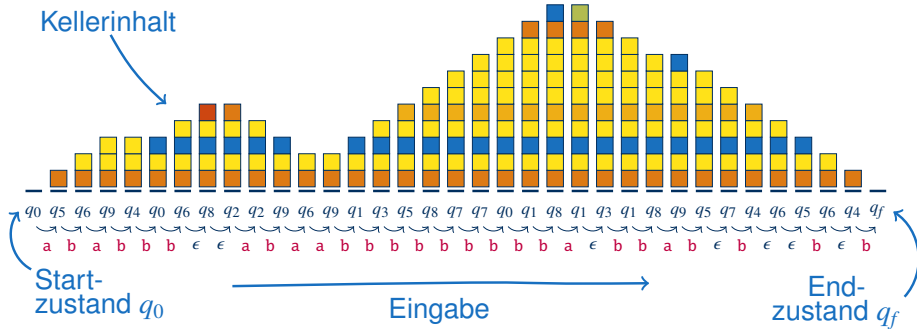
Beweisidee: Für jedes Paar von Zuständen q und r betrachten wir die Sprache $L_{q,r}$, die durch \mathcal{P} akzeptiert wird, wenn man mit leerem Keller in Zustand q beginnt und in Zustand r mit leerem Keller endet, d.h.

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

- $L_{q,r}$ 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 $L_{q,r}$

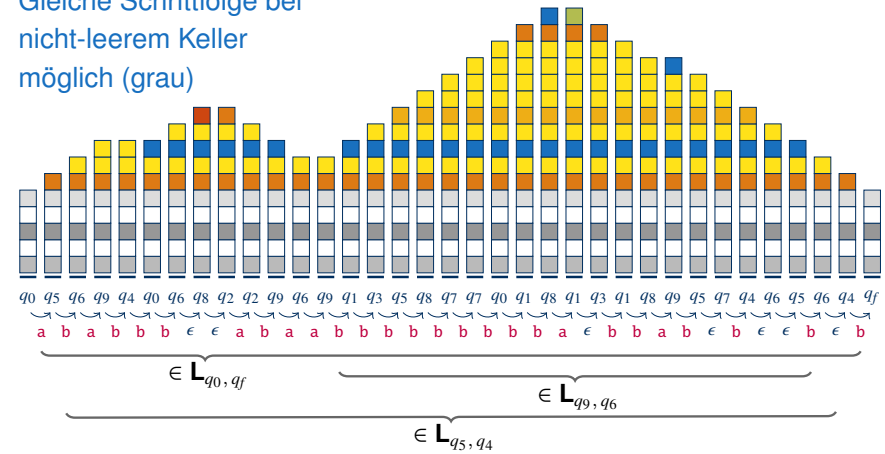
Schematische Darstellung der Konfigurationsfolge bei der Akzeptanz eines Wortes aus L_{q_0, q_f} :



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

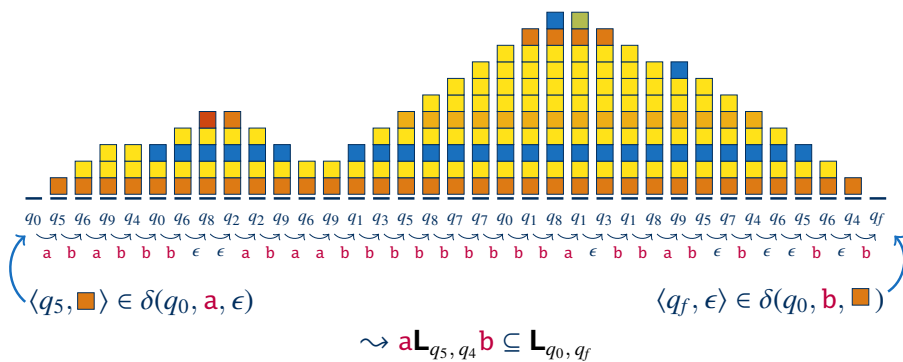
Schematische Darstellung der Konfigurationsfolge bei der Akzeptanz eines Wortes aus L_{q_0, q_f} :

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:

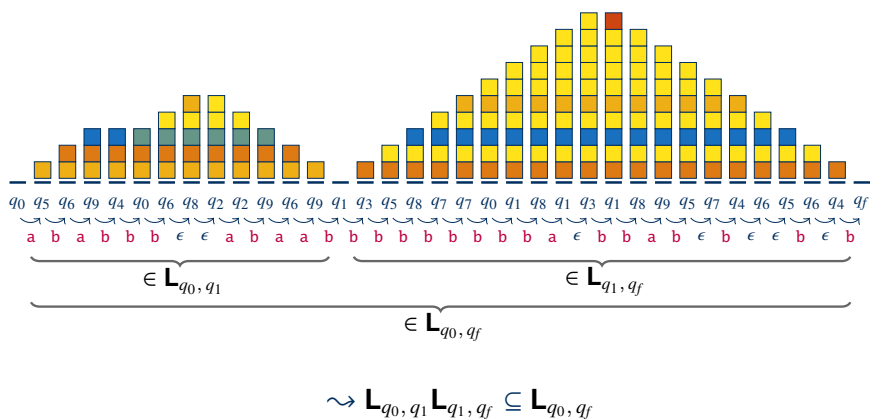
- Wenn es ein Kellersymbol A gibt und
- wenn es ein Symbol a und einen Zustand s_1 gibt, so dass $\langle s_1, A \rangle \in \delta(q, a, \epsilon)$, und
- wenn es ein Symbol b und einen Zustand s_2 gibt, so dass $\langle r, \epsilon \rangle \in \delta(s_2, b, A)$
- dann sollte die Grammatik die Regel

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

enthalten.

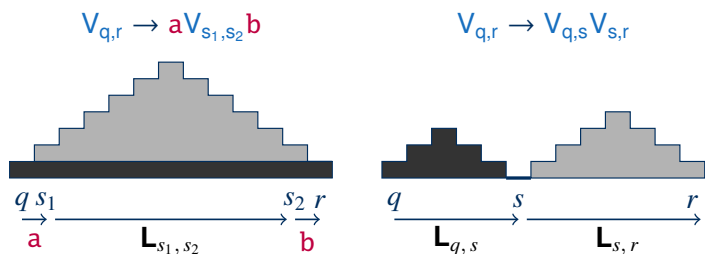
Die Sprache $L_{q,r}$ (4)

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.

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

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

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 Hilfszeichen pusht und diese 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.

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:

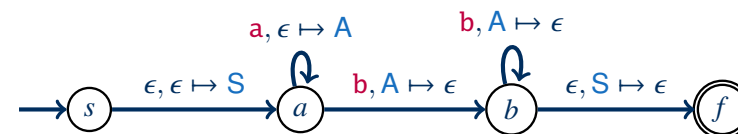
- (1) Für alle $p, q, s_1, s_2 \in Q$, $a, b \in \Sigma_\epsilon$ und $A \in \Gamma$, für die es Übergänge $\langle s_1, A \rangle \in \delta(q, a, \epsilon)$ und $\langle r, \epsilon \rangle \in \delta(s_2, b, A)$ gibt, die Regel $V_{q,r} \rightarrow aV_{s_1,s_2}b$
- (2) Für alle Zustände $p, q, s \in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

Deterministische Kellerautomaten

Beispiel

PDA für $\{a^i b^i \mid i \geq 1\}$:



Dieser PDA erfüllt die Anforderungen aus dem Beweis.

Als Grammatik ergibt sich:

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

(nicht alle dieser Regeln werden wirklich benötigt)

Deterministische Kellerautomaten?

Es gibt verschiedene Quellen für Nichtdeterminismus bei PDAs:

- Übergangsfunktion liefert eine Menge möglicher Übergänge
- Mehrere mögliche Startzustände
- ϵ -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 einelementige Mengen liefert.

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

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**
- Σ : **Eingabealphabet**
- Γ : **Kelleralphabet**
- δ : **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 kontextfreie Sprachen

DPDAs führen zu einer eigenen Sprachklasse:

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

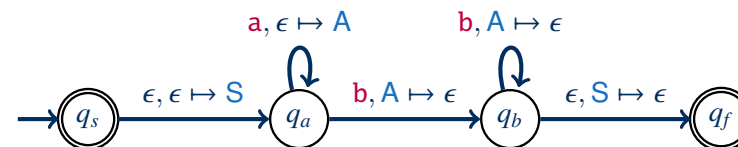
Offensichtlich ist jede deterministisch kontextfreie Sprache auch kontextfrei (DPDAs können als PDAs gesehen werden).

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

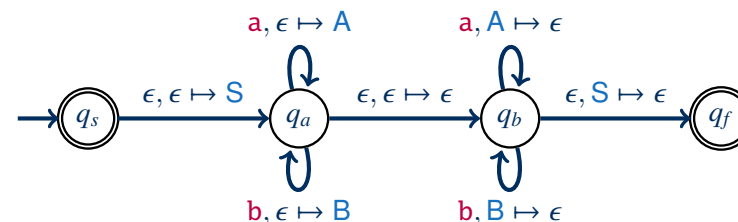
Satz: Die deterministisch kontextfreien Sprachen bilden eine echte Untermenge der kontextfreien Sprachen. Es gibt also kontextfreie Sprachen, die nicht deterministisch sind.

Beispiele

Dieser Automat ist ein DPDA:



Dieser Automat ist **kein** DPDA:



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 komplizierter:

- 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 ϵ -Übergängen gerät. Man muss Grund (2) ausschalten.
- Ein Lauf könnte mit einer Folge von ϵ -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 ϵ -Übergängen akzeptieren darf.

(vollständiger Beweis siehe Sipser, Abschnitt 2.4)



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)

Zusammenfassung und Ausblick

PDA's erlauben **verschiedene äquivalente Definitionen**

PDA's erkennen **genau die kontextfreien Sprachen**:

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

Deterministische Kellerautomaten (DPDA's) 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?