

Grundlagen der Theoretischen Informatik

4. Kellerautomaten und kontextfreie Sprachen (IV)

2.06.2021

Viorica Sofronie-Stokkermans

e-mail: sofronie@uni-koblenz.de

Übersicht

1. Motivation
2. Terminologie
3. Endliche Automaten und reguläre Sprachen
4. Kellerautomaten und kontextfreie Sprachen
5. Turingmaschinen und rekursiv aufzählbare Sprachen
6. Berechenbarkeit, (Un-)Entscheidbarkeit
7. Komplexitätsklassen P und NP

Letzte Vorlesung

Pushdown-Automaten (PDAs)

Push-Down-Automat

Definition(Push-Down-Automat)

Ein **Push-Down-Automat (PDA)** ist ein Tupel

$$\mathcal{M} = (K, \Sigma, \Gamma, \Delta, s_0, Z_0, F)$$

Dabei ist:

K eine endliche Menge von Zuständen

Σ das Eingabealphabet

Γ das Stack- oder Kellularphabet

$s_0 \in K$ der Startzustand

$Z_0 \in \Gamma$ das Anfangssymbol im Keller

$F \subseteq K$ eine Menge von finalen Zuständen

Δ die Zustandsübergangsrelation,
eine endliche Relation: $\Delta \subset (K \times (\Sigma \cup \{\varepsilon\}) \times \Gamma) \times (K \times \Gamma^*)$

Push-Down-Automat

Arbeitsschritt eines PDA

In Abhängigkeit

- vom aktuellen Zustand
- vom nächsten Eingabezeichen (oder auch unabhängig davon)
- vom obersten Kellersymbol

geschieht folgendes

- nächstes **Eingabezeichen** wird **gelesen oder nicht** (bei ε),
- das oberste **Kellersymbol** wird **entfernt**,
- der **Zustand** wird **geändert**,
- es werden null oder mehr **Zeichen auf den Keller** geschoben
Bei neuem Keller-Wort $\gamma = A_1 \dots A_n$ wird A_n zuerst auf den Keller geschoben usw., so dass am Schluss A_1 obenauf liegt.

Push-Down-Automat

Notation

- a, b, c für Buchstaben aus Σ
- u, v, w für Wörter aus Σ^*
- A, B für Stacksymbole aus Γ
- γ, η für Stackinhalte aus Γ^*

PDA: Finaler Zustand / leerer Keller

Definition (von PDA akzeptierte Sprache)

Ein PDA \mathcal{M} kann auf zwei verschiedene Arten eine Sprache akzeptieren:

- über **finale Zustände**
- über **leeren Keller**

$$L_f(\mathcal{M}) = \{w \in \Sigma^* \mid \exists q \in F \exists \gamma \in \Gamma^* ((s_0, w, Z_0) \vdash_{\mathcal{M}}^* (q, \varepsilon, \gamma))\}$$

$$L_l(\mathcal{M}) = \{w \in \Sigma^* \mid \exists q \in K ((s_0, w, Z_0) \vdash_{\mathcal{M}}^* (q, \varepsilon, \varepsilon))\}$$

Bemerkung

Das zu akzeptierende Wort w muss von \mathcal{M} ganz gelesen werden:

$$(s_0, w, Z_0) \vdash^* (q, \varepsilon, \cdot)$$

ist gefordert.

PDA: Finaler Zustand / leerer Keller

Theorem (finale Zustände \rightarrow leerer Keller)

Zu jedem PDA \mathcal{M}_1 existiert ein PDA \mathcal{M}_2 mit

$$L_f(\mathcal{M}_1) = L_l(\mathcal{M}_2)$$

PDA: Finaler Zustand / leerer Keller

Theorem (finale Zustände \rightarrow leerer Keller)

Zu jedem PDA \mathcal{M}_1 existiert ein PDA \mathcal{M}_2 mit

$$L_f(\mathcal{M}_1) = L_l(\mathcal{M}_2)$$

Beweisidee

- Wir simulieren die Maschine \mathcal{M}_1 , die über finale Zustände akzeptiert, durch die Maschine \mathcal{M}_2 , die über leeren Keller akzeptiert.
- \mathcal{M}_2 arbeitet wie \mathcal{M}_1 , mit dem Unterschied:
Wenn ein Zustand erreicht wird, der in \mathcal{M}_1 final war, kann \mathcal{M}_2 seinen Keller leeren.

Details:

<https://userpages.uni-koblenz.de/~sofronie/gti-ss-2016/slides/Endzustand-%3eleeren-Stack.pdf>

Lassen Sie uns nun in die entgegengesetzte Richtung gehen: Wir nehmen einen PDA P_F der eine Sprache L durch Endzustand akzeptiert, und konstruieren einen PDA P_N , der die Sprache L durch Leeren seines Stacks akzeptiert. Die Konstruktion ist einfach und in Abbildung 6.7 dargestellt. Wir fügen zu jedem akzeptierenden Zustand von P_F einen Übergang für die Eingabe ϵ zu einem neuen Zustand p hinzu. Wenn sich P_N im Zustand p befindet, dann leert er seinen Stack und liest kein weiteres Eingabesymbol ein. Daher wird P_N seinen Stack nach dem Einlesen von w leeren, wenn P_F nach dem Einlesen der Eingabe w in einen akzeptierenden Zustand übergeht.

Um zu verhindern, dass eine Situation simuliert wird, in der P_F zufällig seinen Stack leert, ohne zu akzeptieren, muss P_N eine Stackanfangsmarkierung X_0 verwenden. Diese Markierung ist das Startsymbol von P_N und wie bei der Konstruktion von Satz 6.1 muss P_N in einem neuen Zustand p_0 beginnen, dessen einzige Funktion darin besteht, das Startsymbol von P_F auf dem Stack abzulegen und zum Startzustand von P_F zu wechseln. Die Konstruktion ist in Abbildung 6.7 skizziert und wird im nächsten Satz formalisiert.

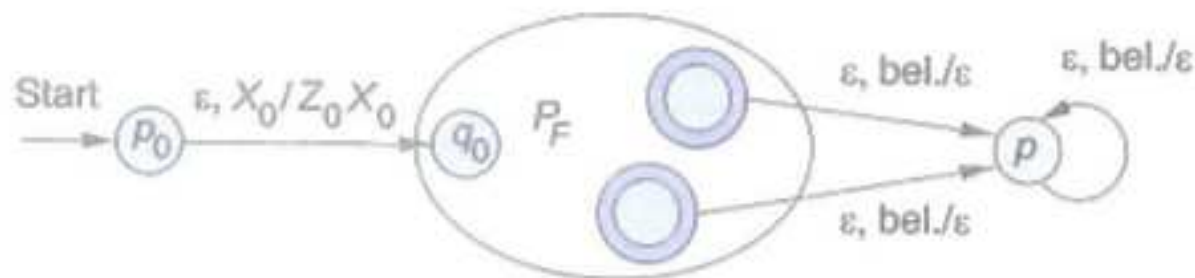


Abbildung 6.7: P_N simuliert P_F und leert seinen Stack genau dann, wenn P_F in einen akzeptierenden Zustand übergeht

PDA: Finaler Zustand / leerer Keller

Theorem (leerer Keller \rightarrow finale Zustände)

Zu jedem PDA \mathcal{M}_1 existiert ein PDA \mathcal{M}_2 mit

$$L_l(\mathcal{M}_1) = L_f(\mathcal{M}_2)$$

PDA: Finaler Zustand / leerer Keller

Theorem (leerer Keller \rightarrow finale Zustände)

Zu jedem PDA \mathcal{M}_1 existiert ein PDA \mathcal{M}_2 mit

$$L_l(\mathcal{M}_1) = L_f(\mathcal{M}_2)$$

Beweisidee

- Wir simulieren die Maschine \mathcal{M}_1 , die über leeren Keller akzeptiert, durch die Maschine \mathcal{M}_2 , die über finale Zustände akzeptiert.
- \mathcal{M}_2 arbeitet wie \mathcal{M}_1 ,
legt aber ein zusätzliches Symbol ganz unten in den Keller.
Wenn \mathcal{M}_1 seinen Keller geleert hätte (also das neue unterste Symbol sichtbar wird), kann \mathcal{M}_2 in einen finalen Zustand gehen.

Details:

<https://userpages.uni-koblenz.de/~sofronie/gti-ss-2016/slides/leeren-Stack-%3eEndzustand.pdf>

Beweis ■ Der diesem Beweis zu Grunde liegende Gedankengang ist in Abbildung 6.4 dargestellt. Wir verwenden ein neues Symbol X_0 , das kein Symbol aus Γ sein darf. X_0 ist sowohl das Startsymbol von P_F als auch eine Markierung am unteren Ende des Stacks, die uns mitteilt, wann P_N einen leeren Stack erreicht hat. Das heißt, wenn P_F X_0 als oberstes Symbol auf dem Stack vorfindet, dann weiß der Automat, dass P_N bei der gleichen Eingabe seinen Stack leeren würde.

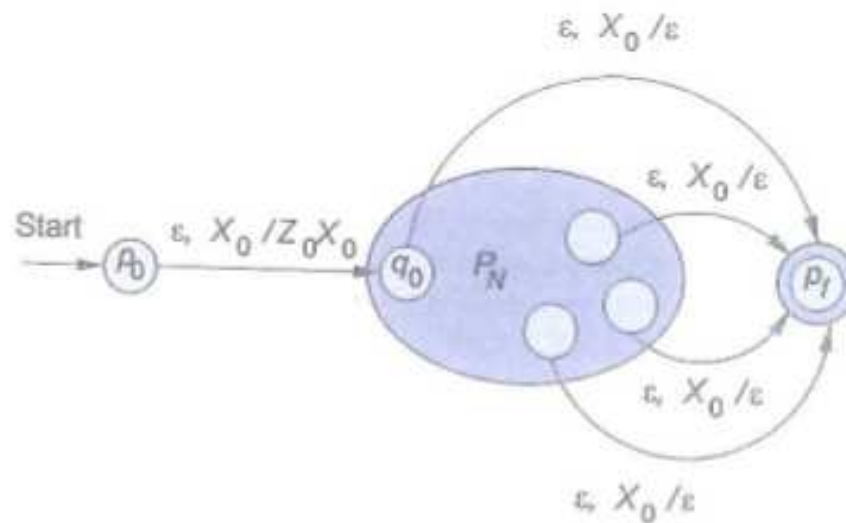


Abbildung 6.4: P_F simuliert P_N und akzeptiert, wenn P_N seinen Stack leert

Wir brauchen auch einen neuen Startzustand p_0 , dessen einzige Funktion darin besteht, das Startsymbol Z_0 von P_N auf dem Stack abzulegen und in den Startzustand q_0 von P_N zu wechseln. Dann simuliert P_F P_N so lange, bis der Stack von P_N leer ist, was P_F daran erkennt, dass er X_0 als oberstes Symbol auf dem Stack vorfindet. Schließlich brauchen wir einen weiteren Zustand p_f , der den akzeptierenden Zustand von P_F repräsentiert. Dieser PDA geht in den Zustand p_f über, sobald er erkennt, dass P_N seinen Stack geleert hat.

Beispiel

Beispiel 6.7 Wir wollen einen PDA entwerfen, der Folgen von if- und else-Wortsymbolen eines C-Programms verarbeitet, wobei i für if und e für else steht. Sie wissen aus Abschnitt 5.3.1, dass es problematisch ist, wenn in einem Präfix mehr else als if vorhanden sind, da es dann nicht möglich ist, jedes else einem voranstehenden if zuzuordnen. Daher werden wir das Stack-Symbol Z einsetzen, um die Differenz zwischen der Anzahl der bereits gelesenen i und der Anzahl der e zu verzeichnen. Dieser einfache PDA, der nur über einen Zustand verfügt, ist im Übergangsdiagramm aus Abbildung 6.5 dargestellt.

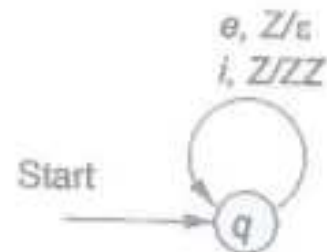


Abbildung 6.5: PDA, der if/e-Fehler durch Leeren des Stacks akzeptiert

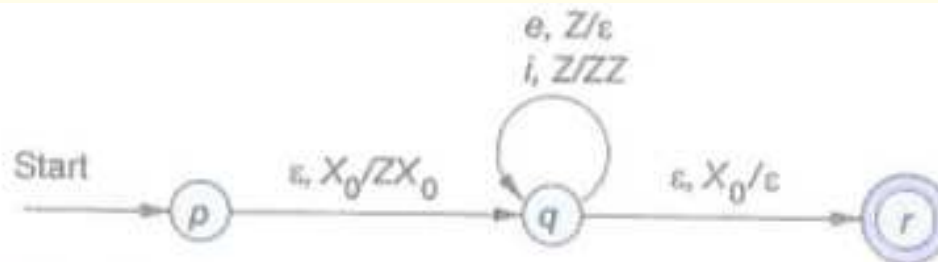


Abbildung 6.6: Konstruktion eines PDA, der durch Endzustand die Sprache des PDA aus Abbildung 6.5 akzeptiert

Lassen Sie uns nun aus P_N einen PDA P_F konstruieren, der dieselbe Sprache durch Endzustand akzeptiert. Das Übergangsdiagramm für diesen PDA P_F ist in Abbildung 6.6³ dargestellt. Wir führen einen neuen Startzustand p und einen akzeptierenden Zustand r ein. Wir werden X_0 als Stackanfangsmarkierung verwenden. Die formale Definition von P_F lautet:

$$P_F = (\{p, q, r\}, \{i, e\}, \{Z, X_0\}, \delta_F, p, X_0, \{r\}),$$

wobei δ_F sich wie folgt zusammensetzt:

- 1. $\delta_F(p, \epsilon, X_0) = \{(q, ZX_0)\}$. Diese Regel bewirkt, dass P_F mit der Simulation von P_N beginnt, wobei X_0 die Stackanfangsmarkierung darstellt.
- 2. $\delta_F(q, i, Z) = \{(q, ZZ)\}$. Diese Regel bewirkt, dass ein Z auf dem Stack abgelegt wird, wenn ein i gelesen wird, entsprechend dem Verhalten von P_N .
- 3. $\delta_F(q, e, Z) = \{(q, \epsilon)\}$. Diese Regel bewirkt, dass ein Z vom Stack entfernt wird, wenn ein e gelesen wird; auch diese Regel simuliert P_N .
- 4. $\delta_F(q, \epsilon, X_0) = \{(r, \epsilon, \epsilon)\}$. Diese Regel bedeutet, dass P_F akzeptiert, wenn der simulierte Automat P_N seinen Stack geleert hat.

Gleichmächtigkeit: PDAs und cf-Grammatiken

Theorem (PDA akzeptieren \mathcal{L}_2)

Die Klasse der PDA-akzeptierten Sprachen ist \mathcal{L}_2 .

Gleichmächtigkeit: PDAs und cf-Grammatiken

Theorem (PDA akzeptieren \mathcal{L}_2)

Die Klasse der PDA-akzeptierten Sprachen ist \mathcal{L}_2 .

Beweis Dazu beweisen wir die folgenden zwei Lemmata, die zusammen die Aussage des Satzes ergeben.

Lemma (cf-Grammatik \rightarrow PDA)

Zu jeder kontextfreien Grammatik G gibt es einen PDA \mathcal{M} mit

$$L(\mathcal{M}) = L(G)$$

Lemma (PDA \rightarrow cf-Grammatik)

Zu jedem Push-Down-Automaten \mathcal{M} gibt es eine kontextfreie Grammatik G mit

$$L(G) = L(\mathcal{M})$$

Gleichmächtigkeit: PDAs und cf-Grammatiken

Lemma (cf-Grammatik \rightarrow PDA) Zu jeder kontextfreien Grammatik G gibt es einen PDA \mathcal{M} mit

$$L(\mathcal{M}) = L(G)$$

Beweis

O.B.d.A. sei die kontextfreie Grammatik $G = (V, T, R, S)$ in Greibach-Normalform: Alle Grammatikregeln haben die Form

$$A \rightarrow au \quad \text{mit } A \in V, a \in T, u \in V^*$$

Wir konstruieren zu G einen PDA \mathcal{M} , der $L(G)$ akzeptiert.

Gleichmächtigkeit: PDAs und cf-Grammatiken

Beweis (Forts.)

Idee: Der Automat \mathcal{M}

- vollzieht die Grammatikregeln nach, die angewendet worden sein könnten, um das aktuelle Eingabewort zu erzeugen und
- merkt sich das aktuelle Wort in der Ableitung bzw. dessen Rest
- merkt sich auf dem Keller alle Variablen, die im gedachten Ableitungswort noch vorkommen und noch ersetzt werden müssen.
- Die linkeste Variable liegt zuoberst: \mathcal{M} arbeitet mit der Linksableitung.

Gleichmächtigkeit: PDAs und cf-Grammatiken

Beweis (Forts.)

Genauer:

- Erzeugung eines Wortes mit G beginnt beim Startsymbol S .
Deshalb S bei \mathcal{M} in Startkonfiguration oben auf dem Keller.

- Angenommen, G hat 2 Regeln mit S auf der linken Seite:
 $S \rightarrow aA_1A_2$ und $S \rightarrow bB_1B_2$

Angenommen, der erste Buchstabe des Input-Wortes w ist ein a .

Wenn w von G erzeugt wurde, hat G die erste der zwei S -Produktionen angewendet.

Entsprechend: Der Automat \mathcal{M} schiebt A_1A_2 auf den Stack.

Gleichmächtigkeit: PDAs und cf-Grammatiken

Beweis (Forts.)

Genauer:

- Der zweite Buchstabe des Eingabeworts muss durch Anwendung einer Regel $A_1 \rightarrow a_1\alpha$ erzeugt worden sein.

Angenommen, der zweite Buchstabe des Eingabeworts ist a_1 . Dann müssen die nächsten Buchstaben des Wortes aus den Variablen in α entstehen.

Der Automat entfernt A_1 vom Stack und legt α auf den Stack.

- Wenn es zwei Regeln $A_1 \rightarrow a_1\alpha_1$ und $A_1 \rightarrow a_1\alpha_2$ gibt, dann wählt \mathcal{M} indeterminiert eine der Regeln aus.
- Der PDA hat nur einen einzigen Zustand und akzeptiert über den leeren Keller.

Gleichmächtigkeit: PDAs und cf-Grammatiken

Beweis (Forts.)

Formal:

$$\mathcal{M} = (K, \Sigma, \Gamma, \Delta, s_0, Z_0, F)$$

mit

$$K := \{s_0\}$$

$$\Sigma := T$$

$$\Gamma := V$$

$$Z_0 := S$$

$$F := \emptyset$$

$$\Delta := \{((s_0, a, A), (s_0, \alpha)) \mid A \rightarrow a\alpha \in R\}$$

Gleichmächtigkeit: PDAs und cf-Grammatiken

Beweis (Forts.)

Damit gilt (Beweis s. Buch):

Es gibt eine Linksableitung $S \Longrightarrow_G^* x\alpha$ mit $x \in T^*$, $\alpha \in V^*$

gdw

\mathcal{M} rechnet $(s_0, x, S) \vdash_{\mathcal{M}}^* (s_0, \varepsilon, \alpha)$

Daraus folgt unmittelbar:

$$L(G) = L_{\ell}(\mathcal{M})$$

Gleichmächtigkeit: PDAs und cf-Grammatiken

Beispiel. Die Sprache

$$L = \{ww^R \mid w \in \{a, b\}^+\}$$

wird generiert von der GNF-Grammatik $G = (\{S, A, B\}, \{a, b\}, R, S)$ mit
 $R = \{ S \rightarrow aSa \mid bSb \mid aa \mid bb \}$

Gleichmächtigkeit: PDAs und cf-Grammatiken

Beispiel. Die Sprache

$$L = \{ww^R \mid w \in \{a, b\}^+\}$$

wird generiert von der GNF-Grammatik $G = (\{S, A, B\}, \{a, b\}, R, S)$ mit
 $R = \{ S \rightarrow aSa \mid bSb \mid aa \mid bb \}$

Grammatik in Greibach Normalform:

$$R' = \{ \begin{array}{l} S \rightarrow aSA \mid bSB \mid aA \mid bB \\ A \rightarrow a \\ B \rightarrow b \end{array} \}$$

Gleichmächtigkeit: PDAs und cf-Grammatiken

Beispiel. Die Sprache

$$L = \{ww^R \mid w \in \{a, b\}^+\}$$

wird generiert von der GNF-Grammatik $G = (\{S, A, B\}, \{a, b\}, R', S)$ mit

$$R' = \{ \begin{array}{l} S \rightarrow aSA \mid bSB \mid aA \mid bB \\ A \rightarrow a \\ B \rightarrow b \end{array} \}$$

Daraus kann man einen PDA mit den folgenden Regeln konstruieren:

$$(s_0, a, S) \Delta (s_0, SA)$$

$$(s_0, a, S) \Delta (s_0, A)$$

$$(s_0, b, S) \Delta (s_0, SB)$$

$$(s_0, b, S) \Delta (s_0, B)$$

$$(s_0, a, A) \Delta (s_0, \varepsilon)$$

$$(s_0, b, B) \Delta (s_0, \varepsilon)$$

Gleichmächtigkeit: PDAs und cf-Grammatiken

Lemma (PDA \rightarrow cf-Grammatik) Zu jedem Push-Down-Automaten \mathcal{M} gibt es eine kontextfreie Grammatik G mit

$$L(G) = L(\mathcal{M})$$

Beweis

Sei \mathcal{M} ein PDA, der eine Sprache L **über leeren Keller** akzeptiert.

Wir konstruieren aus dem Regelsatz von \mathcal{M} eine kontextfreie Grammatik, die L erzeugt.

Gleichmächtigkeit: PDAs und cf-Grammatiken

Beweis (Forts.)

Idee:

Die Variablen der Grammatik sind 3-Tupel der Form

$$[q, A, p]$$

Bedeutung:

Grammatik kann Wort x aus Variablen $[q, A, p]$ ableiten

gdw

\mathcal{M} kann vom Zustand q in den Zustand p übergehen,
dabei A vom Keller entfernen (sonst den Keller unverändert lassen) und
das Wort x lesen:

$$([q, A, p] \Longrightarrow^* x) \quad \underline{\text{gdw}} \quad ((q, x, A\gamma) \vdash^* (p, \varepsilon, \gamma))$$

Gleichmächtigkeit: PDAs und cf-Grammatiken

Beweis (Forts.)

Formale Konstruktion:

Sei

$$\mathcal{M} = (K, \Sigma, \Gamma, \Delta, s_0, Z_0, F)$$

ein PDA.

Daraus konstruiert man die Grammatik

$$G = (V, T, R, S)$$

mit

$$V := \{[q, A, p] \mid q, p \in K, A \in \Gamma\} \cup \{S\}$$

$$T := \Sigma$$

und ...

Gleichmächtigkeit: PDAs und cf-Grammatiken

Beweis (Forts.)

... folgenden Regeln in R :

- $S \rightarrow [s_0, Z_0, q]$ für alle $q \in K$,
- $[q, A, q_{m+1}] \rightarrow a [q_1, B_1, q_2][q_2, B_2, q_3] \dots [q_m, B_m, q_{m+1}]$
für jeden Δ -Übergang $(q, a, A) \Delta (q_1, B_1 \dots B_m)$ und
für jede beliebige Kombination $q_2, \dots, q_{m+1} \in K$,
- $[q, A, q_1] \rightarrow a$
für jeden Δ -Übergang $(q, a, A) \Delta (q_1, \varepsilon)$

Dabei ist $a \in \Sigma \cup \{\varepsilon\}$.

Siehe Buch für Beweis, dass die Konstruktion das gewünschte Ergebnis liefert:

$$([q, A, p] \Longrightarrow^* x) \underline{\text{gdw}} ((q, x, A) \vdash^* (p, \varepsilon, \varepsilon))$$

woraus sofort $L_\ell(\mathcal{M}) = L(G)$ folgt.

Gleichmächtigkeit: PDAs und cf-Grammatiken

Beispiel

Sprache:

$$L_{ab} = \{a^n b^n \mid n \in \mathbb{N}\}$$

L_{ab} wird über leeren Keller akzeptiert von dem PDA

$$\mathcal{M} = (\{s_0, s_1\}, \{a, b\}, \{Z_0, A\}, s_0, Z_0, \emptyset)$$

mit den Regeln

1. $(s_0, \varepsilon, Z_0) \Delta (s_0, \varepsilon)$
2. $(s_0, a, Z_0) \Delta (s_0, A)$
3. $(s_0, a, A) \Delta (s_0, AA)$
4. $(s_0, b, A) \Delta (s_1, \varepsilon)$
5. $(s_1, b, A) \Delta (s_1, \varepsilon)$

Gleichmächtigkeit: PDAs und cf-Grammatiken

Beispiel (Forts.)

Die Transformation ergibt folgende Grammatik-Regeln:

- $$S \rightarrow [s_0, Z_0, s_0] \mid [s_0, Z_0, s_1]$$
1. $[s_0, Z_0, s_0] \rightarrow \varepsilon$
 2. $[s_0, Z_0, s_0] \rightarrow a[s_0, A, s_0]$
 $[s_0, Z_0, s_1] \rightarrow a[s_0, A, s_1]$
 3. $[s_0, A, s_0] \rightarrow a[s_0, A, s_0][s_0, A, s_0]$
 $[s_0, A, s_0] \rightarrow a[s_0, A, s_1][s_1, A, s_0]$
 $[s_0, A, s_1] \rightarrow a[s_0, A, s_0][s_0, A, s_1]$
 $[s_0, A, s_1] \rightarrow a[s_0, A, s_1][s_1, A, s_1]$
 4. $[s_0, A, s_1] \rightarrow b$
 5. $[s_1, A, s_1] \rightarrow b$

Gleichmächtigkeit: PDAs und cf-Grammatiken

Regeln in R :

- $S \rightarrow [s_0, Z_0, q]$ für alle $q \in K$,
- $[q, A, q_{m+1}] \rightarrow a [q_1, B_1, q_2][q_2, B_2, q_3] \dots [q_m, B_m, q_{m+1}]$
für jeden Δ -Übergang $(q, a, A) \Delta (q_1, B_1 \dots B_m)$ und
für jede beliebige Kombination $q_2, \dots, q_{m+1} \in K$,
- $[q, A, q_1] \rightarrow a$
für jeden Δ -Übergang $(q, a, A) \Delta (q_1, \varepsilon)$

Dabei ist $a \in \Sigma \cup \{\varepsilon\}$.

Gleichmächtigkeit: PDAs und cf-Grammatiken

Beispiel (Forts.)

Die Transformation ergibt folgende Grammatik-Regeln:

- $$S \rightarrow [s_0, Z_0, s_0] \mid [s_0, Z_0, s_1]$$
1. $[s_0, Z_0, s_0] \rightarrow \varepsilon$
 2. $[s_0, Z_0, s_0] \rightarrow a[s_0, A, s_0]$
 $[s_0, Z_0, s_1] \rightarrow a[s_0, A, s_1]$
 3. $[s_0, A, s_0] \rightarrow a[s_0, A, s_0][s_0, A, s_0]$
 $[s_0, A, s_0] \rightarrow a[s_0, A, s_1][s_1, A, s_0]$
 $[s_0, A, s_1] \rightarrow a[s_0, A, s_0][s_0, A, s_1]$
 $[s_0, A, s_1] \rightarrow a[s_0, A, s_1][s_1, A, s_1]$
 4. $[s_0, A, s_1] \rightarrow b$
 5. $[s_1, A, s_1] \rightarrow b$

Gleichmächtigkeit: PDAs und cf-Grammatiken

Beispiel (Forts.)

Lesbarer haben wir damit folgende Grammatik:

$$S \rightarrow A \mid B$$

$$A \rightarrow aC \mid \varepsilon$$

$$B \rightarrow aD$$

$$C \rightarrow aCC \mid aDE$$

$$D \rightarrow aCD \mid aDF \mid b$$

$$F \rightarrow b$$

Man sieht jetzt:

- Variable E ist nutzlos, und damit auch die Variable C .
- Die Grammatik enthält Kettenproduktionen und nullbare Variablen.

Gleichmächtigkeit: PDAs und cf-Grammatiken

Beispiel (Forts.)

Nach Entfernung der überflüssigen Elemente:

$$S \rightarrow \varepsilon \mid aD$$

$$D \rightarrow aDF \mid b$$

$$F \rightarrow b$$

Mit dieser Grammatik kann man z.B. folgende Ableitung ausführen:

$$\begin{aligned} S &\Longrightarrow aD \Longrightarrow aaDF \Longrightarrow aaaDFF \Longrightarrow aaabFF \\ &\Longrightarrow aaabbF \Longrightarrow aaabbb \end{aligned}$$

Bis jetzt

- Push-Down-Automaten • Von PDA akzeptierte Sprache:
 - über **finale Zustände**
 - über **leeren Keller**
- Gleichmächtigkeit: PDAs und cf-Grammatiken