

Grundlagen der Theoretischen Informatik

4. Kellerautomaten und kontextfreie Sprachen (V)

7.06.2018

Viorica Sofronie-Stokkermans

e-mail: sofronie@uni-koblenz.de

Organisatorisches

1. Teilklausur: Freitag, 8. Juni 2018, 14:30-15:30, D028

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

Push-Down-Automat

Push-Down-Automat (PDA): Informell

- Wie endlicher Automat, aber **zusätzlich** einen Stack
- Übergangsrelation bezieht das oberste Stacksymbol in den Übergang ein
- Bei Zustandsübergang: lesen und schreiben auf Stack

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^*

Push-Down-Automat: Konfiguration

Konfiguration eines PDA: Informell

- Konfiguration beschreibt die aktuelle Situation des PDA **komplett**
- Bestandteile:
 - **aktueller Zustand**
 - **noch zu lesendes Restwort**
 - **kompletter Stackinhalt**
- Für Konfigurationen C_1, C_2 bedeutet

$$C_1 \vdash C_2$$

dass der PDA in einem Schritt von C_1 nach C_2 gelangen kann.

Push-Down-Automat: Konfiguration

Definition (Konfiguration eines PDA)

Eine **Konfiguration** C eines PDA $\mathcal{M} = (K, \Sigma, \Gamma, \Delta, s_0, Z_0, F)$ ist ein Tripel

$$(q, w, \gamma) \in K \times \Sigma^* \times \Gamma^*.$$

- q der aktuelle Zustand
- w der noch zu lesendes Restwort
- γ der komplette Stackinhalt

Definition (Startkonfiguration)

Bei Eingabewort w ist die **Startkonfiguration**:

$$(s_0, w, Z_0)$$

Push-Down-Automat: Konfiguration

Definition (Nachfolgekonfiguration)

C_2 heißt **Nachfolgekonfiguration** von C_1 ,

$$C_1 \vdash C_2$$

falls $\exists a \in \Sigma \exists A \in \Gamma \exists w \in \Sigma^* \exists \gamma, \eta \in \Gamma^*$ so dass

entweder $C_1 = (q_1, aw, A\gamma)$, $C_2 = (q_2, w, \eta\gamma)$, und $(q_1, a, A) \Delta (q_2, \eta)$,

oder $C_1 = (q_1, w, A\gamma)$, $C_2 = (q_2, w, \eta\gamma)$, und $(q_1, \varepsilon, A) \Delta (q_2, \eta)$.

Definition (Rechnung eines PDA)

Sei \mathcal{M} ein Push-Down-Automat.

$$C \vdash_M^* C'$$

gdw es eine Reihe von Konfigurationen C_0, C_1, \dots, C_n ($n \geq 0$) gibt, so dass:

$$C = C_0, \quad C' = C_n, \quad C_i \vdash_M C_{i+1} \text{ für alle } 0 \leq i < n$$

Dann heißt C_0, C_1, \dots, C_n eine **Rechnung** von \mathcal{M} .

Push-Down-Automat: Akzeptierte Sprache

Definition (von PDA akzeptierte Sprache)

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

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

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

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

Push-Down-Automat: Akzeptierte Sprache

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.

Push-Down-Automat

Bemerkung

- Das unterste Symbol im Keller kann gelöscht werden.
- Dann aber **hängt** der PDA.
- Er kann nicht mehr weiter rechnen.
- **Es gibt keine Nachfolgekonfiguration.**

Push-Down-Automat: Beispiel

Beispiel

Sprache der Palindrome über $\{a, b\}$:

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

L wird über leeren Keller akzeptiert von dem PDA

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

mit ...

Push-Down-Automat: Beispiel

Beispiel (Forts.) Idee:

- Ein Palindrom $w = w^R$ hat die Form

$$vv^R \quad \text{oder} \quad vav^R \quad \text{oder} \quad vbv^R$$

für ein $v \in \{a, b\}^*$

- Der Automat \mathcal{M} liest v und merkt sich jeden Buchstaben.
- **Er rät indeterminiert die Wortmitte.**

Falls das Wort eine ungerade Anzahl von Buchstaben hat,
also $w = vav^R$ oder $w = vbv^R$,
dann muss dabei ein Buchstabe überlesen werden.

- Der Stack enthält nun v^R .
 \mathcal{M} muss jetzt nur noch jeden weiteren gelesenen Buchstaben mit dem jeweils obersten Kellersymbol vergleichen.

Push-Down-Automat: Beispiel

Beispiel (Forts.)

$(s_0, \varepsilon, Z_0) \Delta (s_1, \varepsilon)$	}	ε akzeptieren
$(s_0, a, Z_0) \Delta (s_0, A)$		
$(s_0, a, A) \Delta (s_0, AA)$	}	Stack aufbauen
$(s_0, a, B) \Delta (s_0, AB)$		
$(s_0, b, Z_0) \Delta (s_0, B)$		
$(s_0, b, A) \Delta (s_0, BA)$		
$(s_0, b, B) \Delta (s_0, BB)$		

Push-Down-Automat: Beispiel

Beispiel (Forts.)

$(s_0, \varepsilon, A) \quad \Delta (s_1, \varepsilon)$
 $(s_0, \varepsilon, B) \quad \Delta (s_1, \varepsilon)$ } Richtungswechsel für Palindrome
mit ungerader Buchstabenanzahl

$(s_0, a, A) \quad \Delta (s_1, \varepsilon)$
 $(s_0, b, B) \quad \Delta (s_1, \varepsilon)$ } Richtungswechsel für Palindrome
mit gerader Buchstabenanzahl

$(s_1, a, A) \quad \Delta (s_1, \varepsilon)$
 $(s_1, b, B) \quad \Delta (s_1, \varepsilon)$ } Stack abbauen

Push-Down-Automat: Beispiel

Beispiel (Forts.)

Für das Eingabewort *abbabba* rechnet \mathcal{M} so:

$$\begin{aligned} (s_0, \text{abbabba}, Z_0) \vdash (s_0, \text{bbabba}, A) \vdash (s_0, \text{babba}, BA) \vdash \\ (s_0, \text{abba}, BBA) \vdash (s_0, \text{bba}, ABBA) \vdash (s_1, \text{bba}, BBA) \vdash \\ (s_1, \text{ba}, BA) \vdash (s_1, \text{a}, A) \vdash (s_1, \varepsilon, \varepsilon) \end{aligned}$$

Push-Down-Automat: Beispiel

Die Sprache

$$L = \{w \in \{a, b\}^* \mid \#_a(w) = \#_b(w)\}$$

wird über finalen Zustand akzeptiert von dem PDA

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

mit ...

Push-Down-Automat: Beispiel

Idee:

- auf dem Stack mitzählen, wieviel A -Überhang oder B -Überhang momentan besteht
- Der Stack enthält zu jedem Zeitpunkt
 - entweder nur A/\underline{A} (A -Überhang)
 - oder nur B/\underline{B} (B -Überhang)
 - oder nur das Symbol Z_0 (Gleichstand).
- Das unterste A bzw. B auf dem Stack ist durch einen Unterstrich gekennzeichnet.

So weiß \mathcal{M} , wenn er dies Stacksymbol löscht, dass dann bis zu diesem Moment gleich viel as wie bs gelesen wurden.

Push-Down-Automat: Beispiel

$$\begin{array}{ll} (s_0, a, Z_0) \Delta (s_1, \underline{A}) & (s_0, b, Z_0) \Delta (s_1, \underline{B}) \\ (s_1, a, \underline{A}) \Delta (s_1, A\underline{A}) & (s_1, b, \underline{B}) \Delta (s_1, B\underline{B}) \\ (s_1, a, A) \Delta (s_1, AA) & (s_1, b, B) \Delta (s_1, BB) \\ (s_1, a, \underline{B}) \Delta (s_0, Z_0) & (s_1, b, \underline{A}) \Delta (s_0, Z_0) \\ (s_1, a, B) \Delta (s_1, \varepsilon) & (s_1, b, A) \Delta (s_1, \varepsilon) \end{array}$$

PDA: Finaler Zustand / leerer Keller

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.

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.

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

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

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

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:

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