Automatentheorie

Kellerautomaten

Prof. Dr. Franz-Karl Schmatzer
schmatzf@dhbw-loerrach.de

Literatur

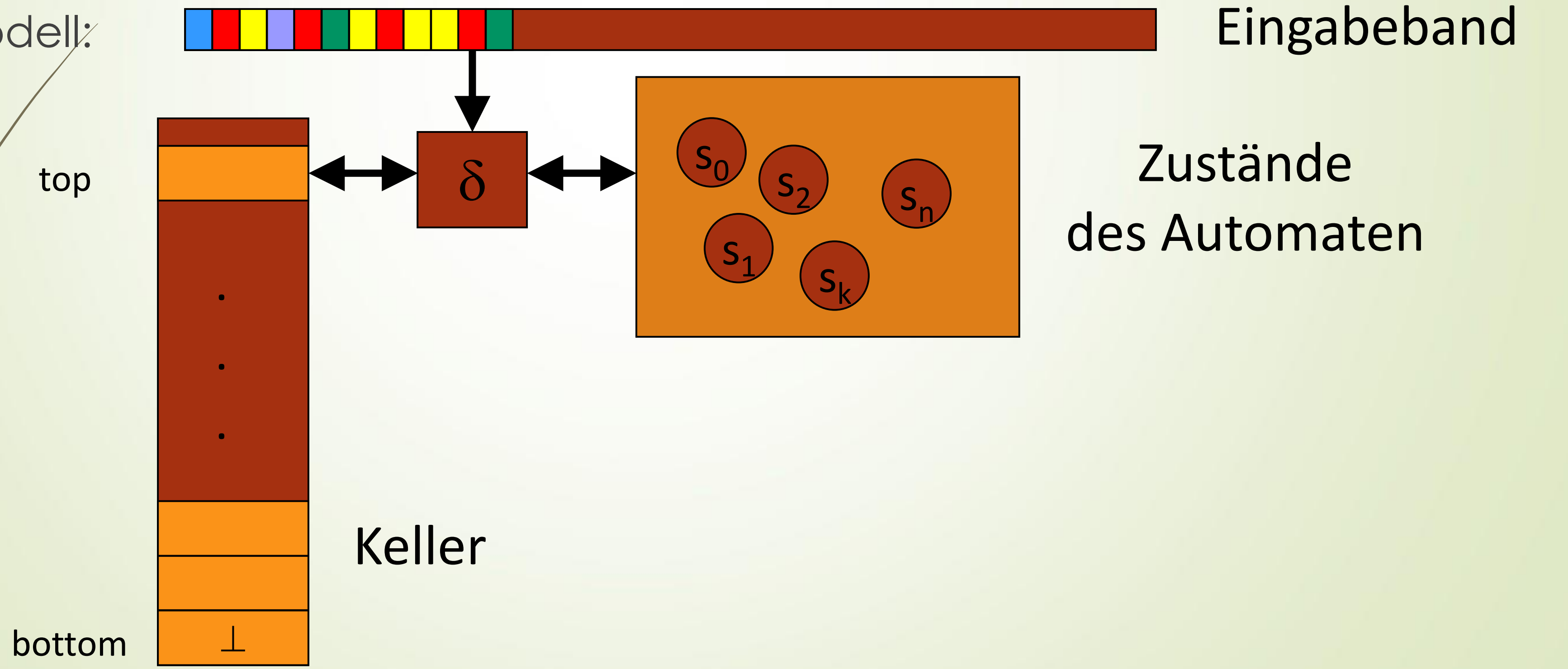
- C.Wagenknecht, M.Hielscher; Formale Sprachen, abstrakte Automaten und Compiler; 3.Aufl. Springer Vieweg 2022;
- Sipser M.; Introduction to the Theory of Computation; 2.Aufl.; Thomson Course Technology 2006
- Hopecroft, T. et al; Introduction to Automata Theory, Language, and Computation; 3. Aufl. Pearson Verlag 2006
- Vossen,G. Witt K.; Grundkurs Theoretische Informatik; 4.Aufl.; Vieweg Verlag 2006
- Cohen, D; Introduction to Computer Theory; John Wiley 1990

Agenda

- Kellerautomaten
- Einführung
  - deterministisch
  - nicht deterministisch

Kellerautomaten
Einführung

- Automaten mit einem unendlichen Speicher mit der Einschränkung:
- Man kann nur auf das oberste Element zugreifen
- Man nennt den Automaten im englischsprachigen Raum Push-Down Automata (PDA)
- Modell:



PDA

Einführung formal

- Sei K = (Σ, Q, K, δ, s_0, k_0, F) ein PDA.
  - $\Sigma = \{e_1,\dots,e_n\}$ eine nicht leere Menge von Zeichen, das Eingabealphabet.
  - $Q = \{s_0,s_1, \quad ,s_n\}$ eine nicht leere Menge von Zuständen.
  - $K = \{k_0,k_1,\dots,k_n\}$ eine nicht leere Menge von Zeichen, das Kelleralphabet.
  - $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times K \rightarrow \wp(Q \times K^*)$ die Überführungsfunktion
  - $s_0 \in S$ der Anfangszustand
  - $k_0 \in$ das Kelleranfangszeichen (oft mit $\perp$ oder \$ bezeichnet)
  - $F \subseteq S$ die nicht leere Menge von Endzustände

PDA

akzeptierte Sprache

- Ein PDA kann auf 2 verschiedenen Weisen eine Sprache akzeptieren.
  - Entweder über finale Zustände $(s_f, \varepsilon, \gamma)$ oder
  - einen leeren Keller $(s, \varepsilon, \varepsilon)$
- Sei $K = (\Sigma, Q, K, \delta, s_0, k_0, F)$ ein PDA. Dann ist

  $L_F(K) = \{w \in \Sigma^* \mid (s_0,w,k_0) \rightarrow^* (s_f, \varepsilon, \gamma), s_f \in F, \gamma \in K^*\}$ bzw.

  $L_L(K) = \{w \in \Sigma^* \mid (s_0,w,k_0) \rightarrow^* (s, \varepsilon, \varepsilon), s \in Q\}$

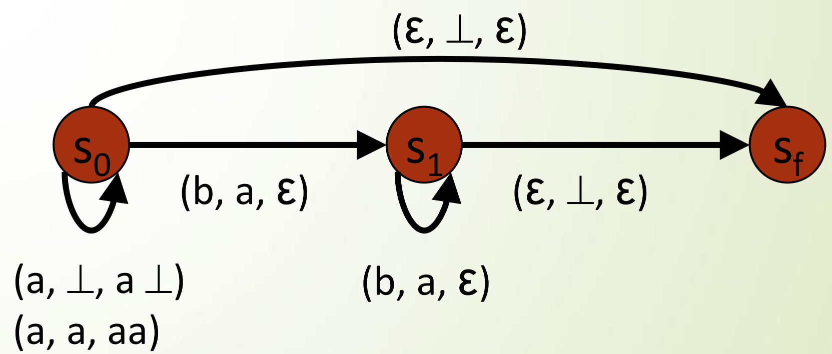
  die von K über S akzeptierte Sprache über finale Zustände $L_F(K)$ bzw. mit einem leeren Keller $L_L(K)$.
- Für einen nicht deterministischen PDA (PDA) sind beide Sprachen äquivalent und man spricht einfach von L(K).
- Für einen deterministischen PDA (DPDA) sind die beiden Sprachen aber nicht äquivalent.
- Mit PDA meinen wir im folgenden immer den nicht deterministischen PDA.

PDA

Arbeitsweise

- Gegeben $L_1 = \{a^n b^n \mid n \geq 0\}$
- Zur Kontrolle der b‘s legen wir die a‘s auf den Kellerspeicher und beim Lesen der b‘s streichen wir für jedes gelesene b ein a. Sind gleichzeitig alle b‘s gelesen und a‘s gestrichen und das Eingabewort abgearbeitet, dann wird das Eingabewort akzeptiert.
- $K_1 = (\{a,b\}, \{s_0,s_1,s_f\}, \{a, \perp\}, \delta_1, s_0, \perp, \{s_f\})$ mit

$$\delta_1 = \{ (s_0, \varepsilon, \perp) \to (s_f, \varepsilon),$$
$$(s_0, a, \perp) \to (s_0, a\perp),$$
$$(s_0, a, a) \to (s_0, aa),$$
$$(s_0, b, a) \to (s_1, \varepsilon),$$
$$(s_1, b, a) \to (s_1, \varepsilon),$$
$$(s_1, \varepsilon, \perp) \to (s_f, \varepsilon)\}$$



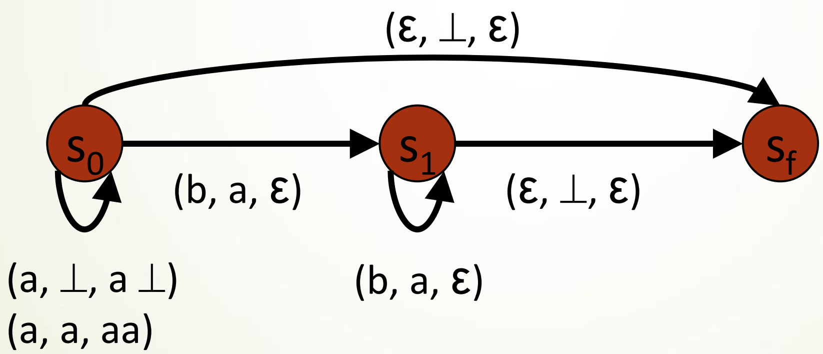
Bauen Sie den Automaten in FLACI nach.

PDA

Arbeitsweise

- Gegeben $L_1 = \{a^n b^n \mid n \geq 0\}$
- Ableiten des Wortes w =aaabbb

$(s_0, aaabbb, \perp) \rightarrow (s_0, aabbb, a\perp) \rightarrow (s_0, abbb, aa\perp) \rightarrow (s_0, bbb, aaa\perp)$
$\rightarrow (s_1, bb, aa\perp) \rightarrow (s_1, b, a\perp) \rightarrow (s_1, \varepsilon, \perp) \rightarrow (s_f, \varepsilon, \varepsilon)$



PDA

Äquivalenz von $L_\varepsilon(K)$ und $L_F(K)$

- Zu zeigen: PDA (finale Zustände ⇔ leerer Keller)
- **(finale Zustände → leerer Keller)**
  - Zu jedem PDA M_1 existiert ein PDA M_2 mit $L_F(M_1) = L_L(M_2)$
  - M_2 soll seinen Keller leeren, falls M_1 in einen finalen Zustand geht. M_2 entsteht aus M_1 durch entsprechende Ergänzung von δ, die den Keller leeren.
- **(leerer Keller → finale Zustände)**
  - Zu jedem PDA M_1 existiert ein PDA M_2 mit $L_L(M_1) = L_F(M_2)$
  - M_2 legt zuerst ein neues unteres Symbol Z_2 in den Keller und arbeitet dann wie M_1. Wenn M_2 das Symbol Z_2 ließt, heißt das M_1 hat den Keller geleert.
  - M_2 geht somit in den finalen Zustand über.
- Die Sprachen $L_L(K)$ und $L_F(K)$ sind gleichmächtig.

PDA

Äquivalenz mit kontextfreien Grammatiken

- Die mit einem PDA akzeptierten Sprachen gehören zur Klasse der kontextfreien Sprachen (Typ-2 Grammatiken)
- Zu jeder kontextfreien Grammatik G lässt sich ein PDA K konstruieren, so dass L(G) = L(K)
- Zu jedem PDA K lässt sich eine kontextfreie Grammatik G angeben, so dass L(K) = L(G)

PDA

Beispiel Palindrome

- Gegeben $L_1 = \{ww^r \mid w \in \{a,b\}^*\}$
- Wir legen die a- und b-Zeichen auf den Stack und probieren immer wieder ab wir schon in der Mitte des Wortes sind und löschen entsprechen.
- $K_1 = (\{a,b\}, \{s_0, s_1, s_f\}, \{a, \perp\}, \delta_1, s_0, \perp, \{s_f\})$ mit

$\delta_1 = \{ (s_0, \varepsilon, \perp) \rightarrow (s_f, \varepsilon), (s_0, a, \perp) \rightarrow (s_0, a\perp),$

$(s_0, a, a) \rightarrow (s_0, aa), (s_0, b, \perp) \rightarrow (s_0, b\perp),$

$(s_0, b, b) \rightarrow (s_0, bb),$

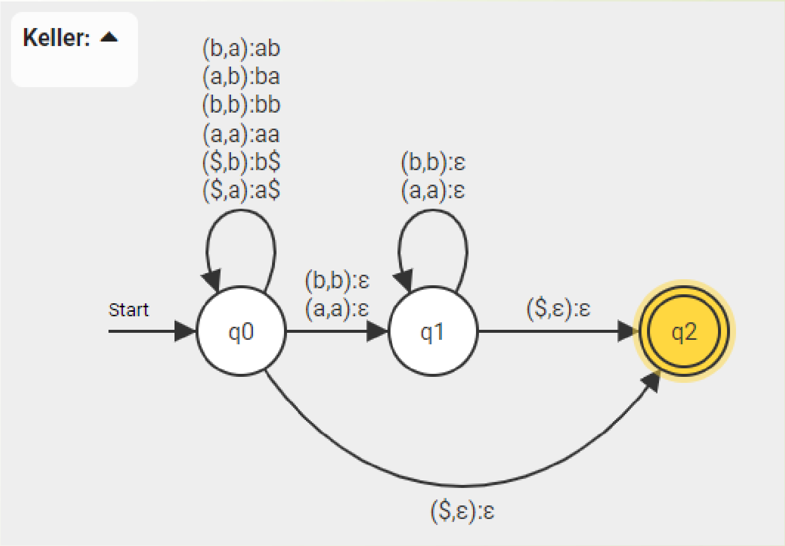
$(s_0, a, b) \rightarrow (s_0, ab),$

$(s_0, b, a) \rightarrow (s_0, ba),$

$(s_0, a, a) \rightarrow (s_1, \varepsilon), (s_0, b, b) \rightarrow (s_1, \varepsilon),$

$(s_1, a, a) \rightarrow (s_1, \varepsilon), (s_1, b, b) \rightarrow (s_1, \varepsilon),$

$(s_1, \varepsilon, \perp) \rightarrow (s_f, \varepsilon)\}$



Bauen Sie den Automaten in FLACI nach.

Aufgabe PDA

- Konstruieren Sie einen PDA, der folgende Sprachen L akzeptieren. (Überprüfen Sie ihr Ergebnis mit FLACI)
  1. $L = \{ 0^n 1^m \mid m > n \geq 0 \}$
  2. $L = \{ w \in \Sigma^* \mid$ Anzahl der 0-Zifferen = Anzahl der 1-Ziffern in w $\}$
  3. $L = \{ 0^n 1^n 2^m \mid n,m \geq 0 \}$
- Konstruieren Sie einen PDA, der die Sprache L der ausgewogenen Klammerausdrücke akzeptiert. D.h. jede öffnende Klammer muss auch eine schließende Klammer haben.

G → PDA

Konstruktion I

Zu jeder kontextfreien Grammatik G = (N ,Σ, P, S) lässt sich ein PDA K_G konstruieren, so dass L(G) = L(K)

- Konstruktion des Automaten K_G (Wir arbeiten das Wort von links her ab)
  - Stimmt das Eingabesymbol mit dem aktuellen Kellersymbol überein. Dann wird das Kellersymbol gelöscht und der Lesekopf auf das nächste Eingabesymbol gesetzt.
  - Ist das Kellersymbol ein Nicht-Terminal A von G, dann wird kein Eingabesymbol gelesen (also ein ε Übergang) und A wird durch die rechte Seite einer Regel von G, deren linke Seite A ist, ersetzt.
  - Das Kellerbottomsymbol ist das Startsymbol S von G.

$G \to PDA$

Konstruktion II

- Der Keller enthält also sowohl Wörter aus Terminalen und Nicht-Terminalen von G und der Kellerautomat hat nur einen Zustand.
- Definition von K_G und δ
  - $K_G = (\Sigma, \{q\}, \Sigma \cup N, \delta, q, S)$
  - $\delta = \{ (q, a, a) \to (q, \varepsilon) \mid a \in \Sigma\} \cup \{(q, \varepsilon, A) \to (q, \alpha) \mid A \to \alpha \in P\}$

$G \rightarrow PDA$

Beispiel1

- Betrachte die Grammatik $G = (\{S\}, \{a, b\}, \{S \rightarrow aSb \mid \varepsilon\}, S)$
  - Diese erzeugt die Sprache $L = \{a^n b^n \mid n \geq 0\}$
- Konstruktion des zugehörigen Automaten K_G (Parser)
  - $K_G = (\{a, b\}, \{q\}, \{a, b, S\}, \delta, q, S)$
  - Die Überführungsfunktion δ setzt sich zusammen aus δ_1 und δ_2
  - $\delta_1 = \{ (q, a, a) \rightarrow (q, \varepsilon), (q, b, b) \rightarrow (q, \varepsilon) \}$ (Terminale Regeln)
  - $\delta_2 = \{ (q, \varepsilon, S) \rightarrow (q, aSb), (q, \varepsilon, S) \rightarrow (q, \varepsilon)\}$ (Nichtterminale Regeln)
- Arbeitsweise:

  $(q, aaabbb, S) \rightarrow (q, aaabbb, aSb) \rightarrow (q, aabbb, Sb) \rightarrow$

  $(q, aabbb, aSbSb) \rightarrow (q, abbb, SbSb) \rightarrow (q, abbb, aSbSbSb) \rightarrow$

  $(q, bbb, SbSbSb) \rightarrow (q, bbb, bSbSb) \rightarrow (q, bb, SbSb) \rightarrow (q, bb, bSb) \rightarrow$

  $(q, b, Sb) \rightarrow (q, b, b) \rightarrow (q, \varepsilon, \varepsilon)$

$G \rightarrow PDA$

Beispiel2

- Betrachte die Grammatik der arithmetischen Ausdrücke
- $G_A = (\{S,O\}, \{a,(,),+,-,*,/\}, P,S)$ und

  $P = \{S \rightarrow a \mid S\ O\ S \mid (S) \mid ,\ O \rightarrow + \mid - \mid * \mid / \}$
- Konstruktion des zugehörigen Automaten K_A (Parser)
  - $K_A = (\{a,(,),+,-,*,/\}, \{q\}, \{a,(,),+,-,*,/, S, O\}, \delta, q, S)$
  - Die Überführungsfunktion δ setzt sich zusammen aus δ_1 und δ_2
  - $\delta_1 = \{ (q, a, a) \rightarrow (q, \varepsilon), (q, (, () \rightarrow (q, \varepsilon), (q,),)) \rightarrow (q, \varepsilon),$

    $(q, +,+) \rightarrow (q, \varepsilon), (q, -, -) \rightarrow (q, \varepsilon), (q, *, *) \rightarrow (q, \varepsilon),$

    $(q, /, /) \rightarrow (q, \varepsilon) \}$
  - $\delta_2 = \{ (q, \varepsilon, S) \rightarrow (q, a), (q, \varepsilon, S) \rightarrow (q, S\ O\ S), (q, \varepsilon, S) \rightarrow (q, (S)),$

    $(q, \varepsilon, O) \rightarrow (q, +), (q, \varepsilon, O) \rightarrow (q, -), (q, \varepsilon, O) \rightarrow (q,*),$

    $(q, \varepsilon, O) \rightarrow (q, /)\}$

G → PDA

Beispiel2

- Arbeitsweise:

  (q, (a+a)*a, S) → (q, (a+a)*a, SOS) → (q, (a+a)*a, (S)OS) →

  (q, a+a)*a,S)OS) → (q, a+a)*a,SOS)OS) → (q, a+a)*a,aOS)OS) →

  (q, +a)*a,OS)OS) → (q, +a)*a,+S)OS) → (q, a)*a,S)OS) →

  (q, a)*a,a)OS) → (q,)*a,)OS) → (q, *a,OS) → (q, *a,*S) →

  (q, a, S) → (q, a, a) → (q, ε, ε)

Aufgabe G-> PDA

- Erstellen Sie für folgende Grammatiken einen PDA, der die Sprache mit leerem Keller und mit finalen Endzustände akzeptiert.
  - $G = (\{S,A\}, \{0,1\}, \{S \to 0S1 \mid A , A \to 1A0 \mid \varepsilon\}, S)$

PDA $\rightarrow$ G

Beweisidee 1

- Zu jedem PDA $K = (\Sigma, Q, K, \delta, s_0, \perp, F)$ lässt sich eine kontextfreie Grammatik $G = (N, \Sigma, P, S)$ angeben, so dass $L(K) = L(G)$
- Beweisidee:
  - Damit ein Wort akzeptiert wird, muss jedes Symbol aus dem Keller genommen werden.
  - Pro δ-Schritt, kann der Keller höchstens um 1 Zeichen schrumpfen.
  - Für jedes Symbol, das auf den Keller gelegt wird, muss dazu ein Buchstaben gelesen oder ein ε-Übergang gemacht werden.

PDA $\rightarrow$ G

Beweisidee 2

- Die Variablen der Grammatik werden als 3-Form [pAq] notiert. Die Interpretation ist: Der PDA kann von Zustand p nach q übergehen und dabei A aus dem Kellerspeicher entfernen.
- Was machen wir mit $(p, a, A) \rightarrow (q, B_1B_2\ldots B_M)$?
- Wir übersetzen das in $[pAq_M] \rightarrow a[qB_1q_1]\ [q_1B_2q_2]\ldots\ [q_{M-1}B_Mq_M]$ für alle Kombinationen von Zuständen $q_1,..q_M \in Z$
- Das heißt um A vom Keller zu entfernen, geht der PDA vom Zustand p in möglicherweise mehren Schritten nach q_M. Im ersten Schritt liest er a und geht zu q. Anschließend schiebt er die Symbole B_1 bis B_M auf den Keller, die er in mindestens M weiteren Schritten wieder abarbeiten muss.

PDA $\to$ G

Konstruktion

- Konstruktion der Grammatik $G = (N, \Sigma, P, S)$ zu

  $K = (\Sigma, Q, K, \delta, s_0, \perp, F)$

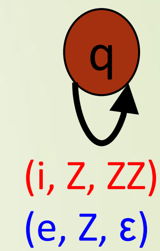
  - Ein spezielles Startsymbol S
  - Alle Symbole N in G haben die Form [pXq], wobei p und q Zustände sind und X ein Kellersymbol sein soll.
- $N = \{S\} \cup \{ [pXq] \mid p,q \in Q , X \in K\}$
- Erstellen der Produktionsregeln P mit $a \in \Sigma \cup \{\varepsilon\}$ und $A \in K$
  - $S \to [s_0 \perp q]$ für alle $q \in Q$ (1)
  - $[pAq] \to a$ für alle δ Übergänge $(p, a, A) \to (q, \varepsilon)$ (2)
  - $[pAq_M] \to a[qB_1q_1]\ [q_1B_2q_2] \ldots\ [q_{M-1}B_Mq_M]$ für alle Kombinationen von q_1 , $q_M \in Q$ und für alle δ Übergänge $(p, a, A) \to (q, B_1B_2 \ldots B_M)$ (3)
- Man kann nun zeigen dass:

  $([pAq] \Rightarrow^* x) \Leftrightarrow ((p, x, A) \to^* (q, \varepsilon, \varepsilon))$

PDA → G

Beispiel1

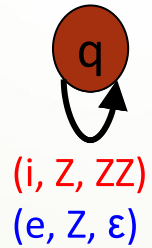
- Es sein PDA P_N = ({i,e}, {q}, {Z}, δ, q, Z) mit der Bedingung: Akzeptieren des Wortes mit leerem Keller.
- Welche Sprache erzeugt P_N?
- Konstruktion der Grammatik G_N
- Produktionsregel
  - (1) S → [q Z q] (die einzige Regel, da nur einen Zustand. Falls n-Zustände, dann würden hier n-Regeln stehen)
  - (2) [q Z q] → e (da δ für e eine Ableitung nach (q, ε) hat)
  - (3) [q Z q] → i[q Z q] [q Z q] (da δ für i eine Ableitung (q, ZZ) enthält. Wieder hat man nur eine Regel, da nur ein Zustand vorliegt. Bei n-Zuständen, erwartet man n*n Regeln.



PDA $\rightarrow$ G

Beispiel1f

- Setzt man A = [q Z q] dann erhält man:
  - P = { $S \rightarrow A$, $A \rightarrow iAA \mid e$}
- S kann man noch entfernen und dann A durch S ersetzen, die Grammatik ist dann : $G_N = (\{S\}, \{i,e\}, \{S \rightarrow iSS \mid e\}, S)$



PDA $\rightarrow$ G

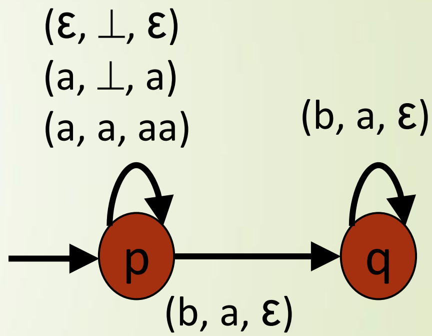
Beispiel2

- Die Sprache $L = \{a^n b^n \mid n \geq 0\}$ wird von einem PDA K_L mit leerem Keller akzeptiert.

  $K_L = (\{a,b\}, \{p,q\}, \{a, \perp\}, \delta_1, p, \perp)$ mit

  $\delta_L = \{ (p, \varepsilon, \perp) \rightarrow (p, \varepsilon), (p, a, \perp) \rightarrow (p, a), (p, a, a) \rightarrow (p, aa),$

  $(p, b, a) \rightarrow (q, \varepsilon), (q, b, a) \rightarrow (q, \varepsilon)\}$



- Konstruktion der Grammatik G_L
- Produktionsregel:
  - (1) $S \rightarrow [p \perp p] \mid [p \perp q]$ (2 Regeln, da 2 Zustände)
  - (2) $[p \perp p] \rightarrow \varepsilon$, $[p\,a\,q] \rightarrow b$, $[q\,a\,q] \rightarrow b$ $\qquad (A \rightarrow a)$
  - (3) $[p \perp p] \rightarrow a\,[p\,a\,p]$, $[p \perp q] \rightarrow a\,[p\,a\,q]$ $\qquad (A \rightarrow aB)$
  - (3‘) $[p\,a\,p] \rightarrow a\,[p\,a\,p]\,[p\,a\,p] \mid a\,[p\,a\,q]\,[q\,a\,p]$ $\qquad (A \rightarrow aBC)$

    $[p\,a\,q] \rightarrow a\,[p\,a\,p]\,[p\,a\,q] \mid a\,[p\,a\,q]\,[q\,a\,q]$

PDA $\rightarrow$ G

Beispiel2

- Setzt man Abkürzungen ein:

  A: = [p⊥p] ,B: = [p⊥q] ,C: = [pap] ,D:= [paq], E: = [qap] ,F:= [qaq]

- so erhält man:

$S \rightarrow A \mid B$	$S \rightarrow \varepsilon \mid aC \mid aD$	$S \rightarrow \varepsilon \mid aD$	$S \rightarrow \varepsilon \mid aD$
$A \rightarrow aC \mid \varepsilon$	-	-	-
$B \rightarrow aD$	-	-	-
$C \rightarrow aCC \mid aDE$	$C \rightarrow aCC$	-	-
$D \rightarrow aCD \mid aDF \mid b$	$D \rightarrow aCD \mid aDb \mid b$	$D \rightarrow aDb \mid b$	$D \rightarrow aDb \mid b$
$F \rightarrow b$	-	-	-

- Entfernen der nutzlose Variablen E und der Kettenproduktionen A,B und F.
- Entfernen der Variable C und zugehörige Produktionen. Denn diese Regeln terminieren nicht.
- $P = \{S \rightarrow \varepsilon \mid aD, D \rightarrow aDb \mid b\}$

PDA $\rightarrow$ G

Beispiel 2

- Die Sprache $L = \{a^n b^n \mid n \geq 0\}$ wird von einem PDA K_F mit finalem Endzustand akzeptiert

$K_F = (\{a,b\}, \{s_0, s_1, s_f\}, \{a, \perp\}, \delta_1, s_0, \perp, \{s_f\})$ mit

$\delta_F = \{ (s_0, \varepsilon, \perp) \rightarrow (s_f, \varepsilon),$

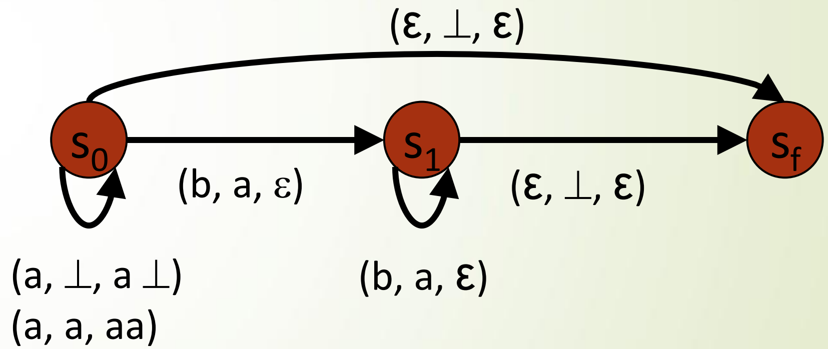
$(s_0, a, \perp) \rightarrow (s_0, a\perp),$

$(s_0, a, a) \rightarrow (s_0, aa),$

$(s_0, b, a) \rightarrow (s_1, \varepsilon),$

$(s_1, b, a) \rightarrow (s_1, \varepsilon),$

$(s_1, \varepsilon, \perp) \rightarrow (s_f, \varepsilon)\}$



- Konstruktion der Grammatik

(Notation $[s_i \perp s_j] = S_{ij}$ und $[s_i\, a\, s_j] = A_{ij}$)

- $S \rightarrow S_{00} \mid S_{01} \mid S_{0f}$ (1) Startregeln
- $S_{0f} \rightarrow \varepsilon$, $A_{01} \rightarrow b$, $A_{11} \rightarrow a$, $S_{1f} \rightarrow \varepsilon$ (2) Terminierungsregel

PDA $\rightarrow$ G

Beispiel2

- $S_{00} \rightarrow aA_{00}S_{00} \mid aA_{01}S_{10} \mid aA_{0f}S_{f0}$
- $S_{01} \rightarrow aA_{00}S_{01} \mid aA_{01}S_{11} \mid aA_{0f}S_{f1}$
- $S_{0f} \rightarrow aA_{00}S_{0f} \mid aA_{01}S_{1f} \mid aA_{0f}S_{ff}$
- $A_{00} \rightarrow aA_{00}A_{00} \mid aA_{01}A_{10} \mid aA_{0f}A_{f0}$
- $A_{01} \rightarrow aA_{00}A_{01} \mid aA_{01}A_{11} \mid aA_{0f}A_{f1}$
- $A_{0f} \rightarrow aA_{00}A_{0f} \mid aA_{01}A_{1f} \mid aA_{0f}A_{ff}$

- Beobachtung: Es werden unnötig viele Variablen erzeugt, die nicht benötigt werden.
- Bereinigung der unnötigen Variablen liefert:

$G = (\{S, S_{0f}, S_{1f}, A_{01}, A_{11}\}, \{a,b\}, P, S)$

mit P: = $\{ S \rightarrow S_{0f},\ S_{0f} \rightarrow a\, A_{01}\, S_{1f} \mid \varepsilon,\ A_{01} \rightarrow a\, A_{01}\, A_{11} \mid b,\ A_{11} \rightarrow b,\ S_{1f} \rightarrow \varepsilon \}$

Vereinfachen zu ($S \rightarrow S_{0f}$ streichen und S_{0f} durch S ersetzen, $A_{11} \rightarrow b$, $S_{1f} \rightarrow \varepsilon$ substituieren und A_{01} durch D ersetzen)

P: = $\{ S \rightarrow aD \mid \varepsilon,\ D \rightarrow aDb \mid b\}$

Aufgaben PDA

- Erstellen Sie für folgende Grammatiken ein PDA, der die Sprache mit leerem Keller akzeptiert.

  $G = (\{S,A\}, \{0,1\}, \{S \to 0S1 \mid A, A \to 1A0 \mid S \mid \varepsilon\}, S)$

  $G = (\{S,A\}, \{a\}, \{S \to aAA, A \to aS \mid a\}, S)$

- Erstellen Sie ein PDA, der folgende reguläre Sprachen mit leerem Keller akzeptiert
  - $\{a^n b^m c^{2(n+m)} \mid n \geq 0, m \geq 0\}$
  - $\{a^i b^j c^k \mid i = 2j \text{ oder } j=2k\}$
- Erstellen sie für den folgenden PDA eine kontextfreie Grammatik
  - $K = (\{0,1\}, \{p,q\}, \{X, \perp\}, \delta, p, \perp)$
  - $\delta(q,1, \perp) = \{(q, X\perp)\}, \delta(q,1, X) = \{(q, XX)\},$
  - $\delta(q,0, X) = \{(p, X)\}, \delta(q, \varepsilon, X) = \{(q, \varepsilon)\},$
  - $\delta(p,1, X) = \{(p, \varepsilon)\}, \delta(p,0, \perp) = \{(q, \perp)\},$

Deterministischer PDA (DPDA)

Einführung

- Die Algorithmen für Wort und Syntaxanalysen (Parser und Scanner) benötigen für eine kontextfreie Grammatik $O(n^3)$ Rechenzeit und sind daher zu langsam
- Die Klasse der deterministischen PDA's erlauben Algorithmen zur Syntaxanalyse anzugeben, die linear $O(n)$ sind.
- Dazu gehören auch die Klassen der deterministischen kontextfreien Grammatiken.

Deterministischer PDA (DPDA)

- Im Gegensatz zu den endlichen Automaten besteht ein Unterschied zwischen deterministischen und nicht deterministischen Kellerautomaten.
- Die Klasse der deterministischen Kellerautomaten L_{DPDA} ist eine echte Teilmenge der Klasse der nicht deterministischen Kellerautomaten L_{PDA} d.h.

$$L_{DPDA} \subset L_{PDA}$$

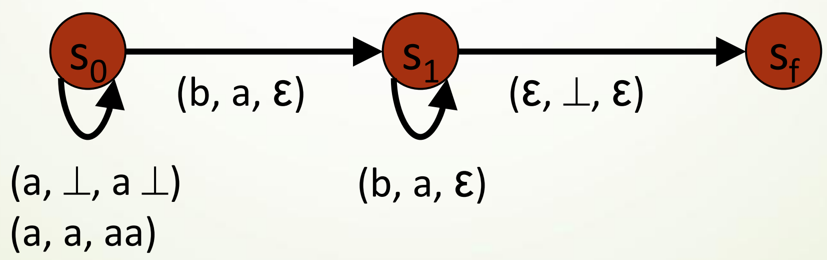
- Z.B Die Sprache der Palindrome gehört zu L_{PDA} nicht aber zu L_{DPDA}
- Weiterhin bestehen auch Unterschiede innerhalb der deterministischen Automaten
  - akzeptieren mit leerem Keller oder
  - akzeptieren mit finalen Zuständen

Deterministischer PDA (DPDA)

- Ein PDA $K = (\Sigma, Q, K, \delta, s_0, k_0, F)$ heißt determiniert, falls gilt:
  - $\delta(q, a, X) \neq \emptyset \Rightarrow \delta(q, \varepsilon, X) = \emptyset$;

    d.h. falls in einem Zustand q mit Kellersymbol X ein Zeichen a gelesen werden kann, darf das es nicht gleichzeitig einen ε -Übergang zu q und X geben.
  - $|\delta(q, a, X)| \leq 1$ für alle $q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $X \in K$;

    d.h. es kann in keiner Konfiguration mehr als einen möglichen Übergang geben.
  - Beispiel $L = \{a^n b^n \mid n \geq 1\}$



DPDA

Sprache eines DPDA

- Deterministische PDAs mit finalen Zuständen sind mächtiger als deterministische PDAs mit leerem Keller.
  - Jeder DPDA mit leerem Keller kann, wie zuvor im Beweis von PDAs in einen DPDA mit finalen Zuständen transformiert werden.
  - L = {a, ab} kann mit einen DPDA nur mit finalen Zuständen akzeptiert werden. Es gibt kein DPDA der dies mit leerem Keller tut.
- L_L(DPDA): die Sprache L, die eine deterministischer Automat (DPDA) mit leerem Keller (L) akzeptiert.
- L_F(DPDA): die Sprache L, die eine deterministischer Automat (DPDA) mit finalen Zuständen (F) akzeptiert.

Aufgabe deterministische Kellerautomaten

- Erstellen Sie ein DPDA, der folgende kontextfreie Sprachen mit leerem Keller akzeptiert
  - $\{a^i b^j \mid i = 2j \text{ und } i,j \geq 1\}$
  - $\{a^n b^m c^{2(n+m)} \mid n,m \geq 1\}$
  - Erstellen Sie diesen Automaten auch in FLACI

DPDA

Präfixeigenschaft

- Sei $L \subseteq \Sigma^*$.
- L hat die Präfixeigenschaft genau dann, wenn für alle $w \in L$ gilt:
  - Ist x ein echter Präfix von w = xz, so ist $x \notin L$
- Beispiel:
  - $L = \{a^n b^n \mid n > 0\}$ hat die Präfix-Eigenschaft
    - zu $w=a^n b^n$ sind alle echten Präfixe von der Form

      $x = a^i$ mit $0 < i \leq n$ oder $x= a^n b^i$ mit $n > i$ und $x \notin L$
  - $L = \{a^i b^j \mid i,j \geq 0\}$ hat nicht die Präfix-Eigenschaft
    - Zu $w = aaabb \in L$ ist x=aa ein Präfix und $x \in L$
- Die Sprache L_L(DPDA) ist genau die Sprache, welche die Präfixeigenschaften haben.

DPDA

Sprachklassen