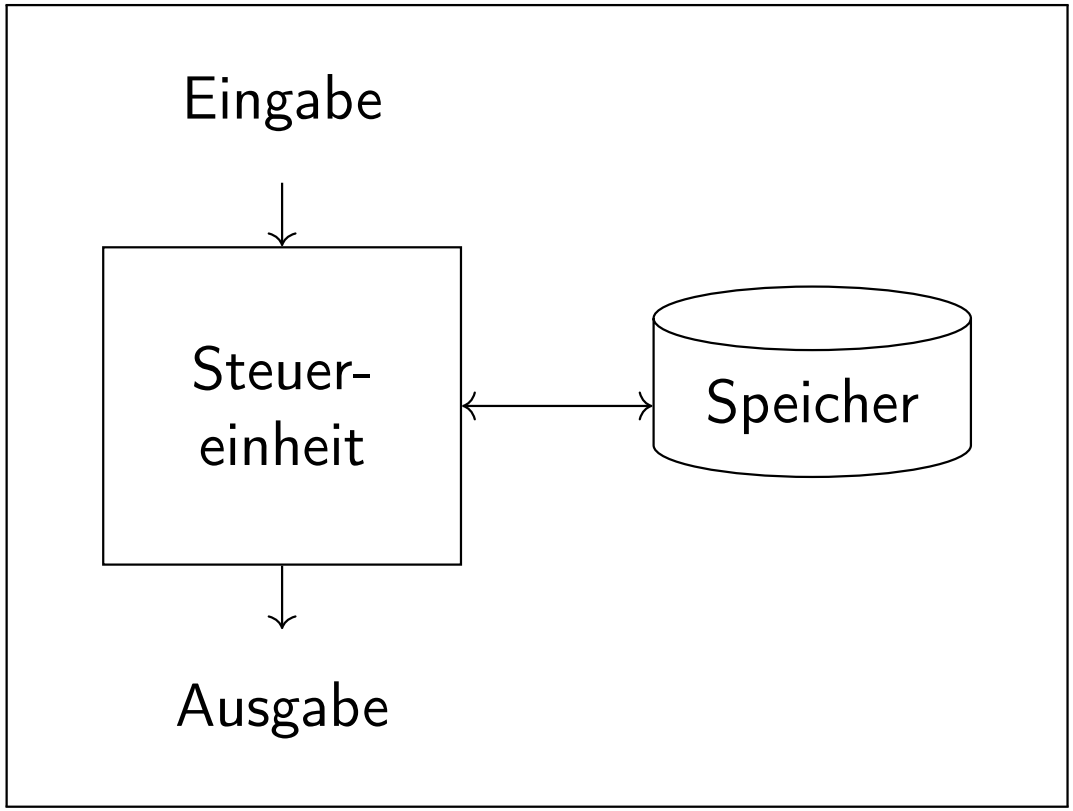
Um ein Automatenmodell für kontextfreie Sprachen zu erhalten,

- führen wir daher einen Keller oder Pushdown-Speicher ein, auf dem sich eine beliebig lange Sequenz von Zeichen befinden darf.
- Beim Einlesen eines neuen Zeichens wird das oberste Zeichen des Kellers gelesen und durch eine (evtl. leere) Sequenz von Zeichen ersetzt.
  An anderen Stellen kann der Keller nicht gelesen oder verändert werden.

# Maschinen: Kellerautomaten




Steuereinheit:

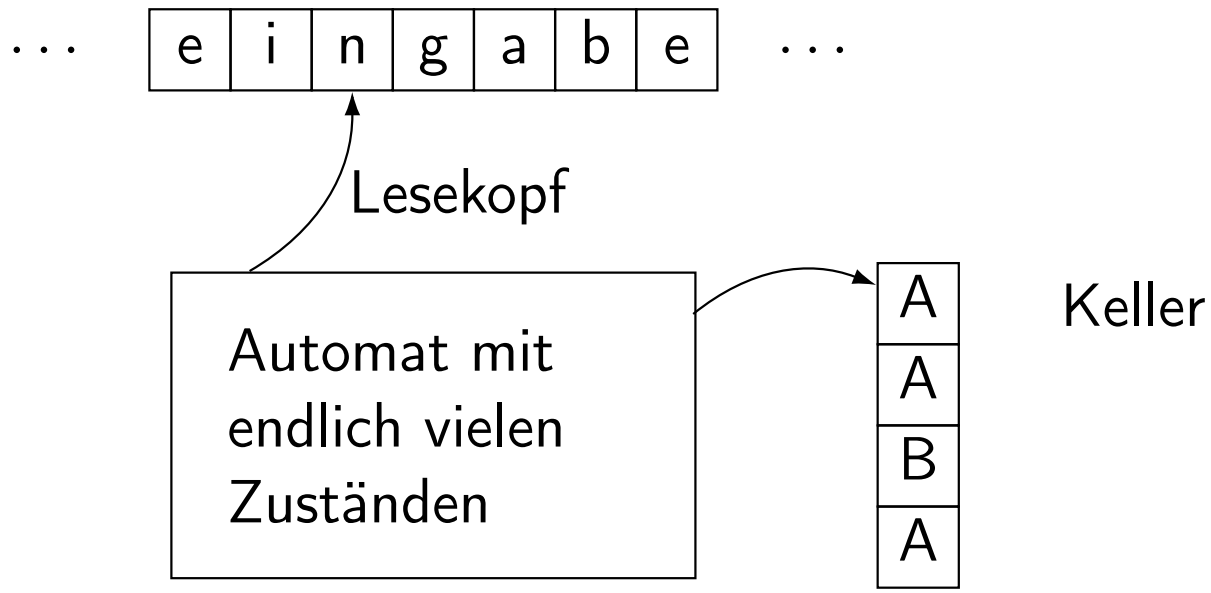
- endlich viele Zustände
- ändern sich in Abhängigkeit von Eingabe und Speicherkonfiguration
- produziert Ausgaben
- gibt „Speicherbefehle“

Eingabe: Folge von „Buchstaben“

Ausgabe: Folge von „Buchstaben“ oder gut/schlecht bzw. 1/0

- Art des Speicherzugriffs: verboten Kellerspeicher Turing-Band
- Art der Steuereinheit: deterministisch nichtdeterministisch
- Dauer der Berechnung: polynomiell exponentiell ...
- Größe des Speichers: fest logarithmisch polynomiell ...

Schematische Darstellung eines Kellerautomaten:

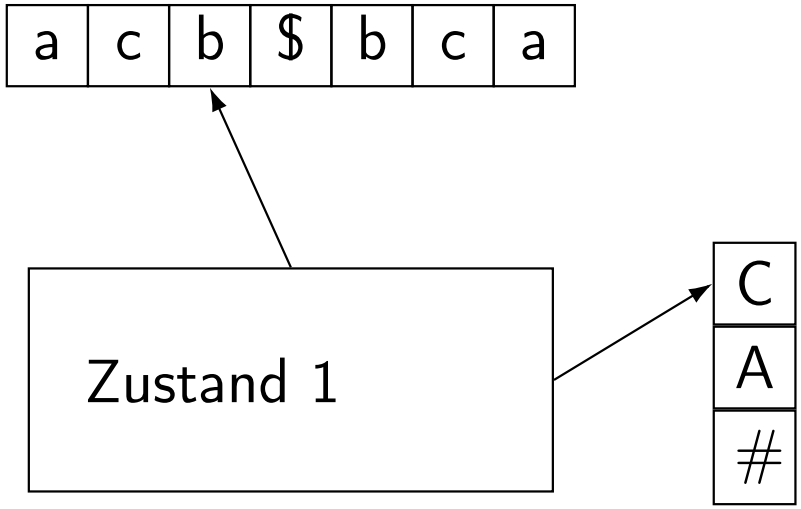


Sei $\Delta = \{a, b, c, d\}$ und
$L = \{a_1 a_2 \cdots a_n \$ a_n \cdots a_2 a_1 \mid n \in \mathbb{N}, a_1, \ldots, a_n \in \Delta\}$.

Ein Kellerautomat erkennt diese Sprache folgendermaßen:

- Ein Wort $w$ wird von links nach rechts eingelesen, wobei am Anfang nur das Zeichen # im Keller steht.
- Solange \$ noch nicht erreicht ist, wird jedes eingelesene Symbol als Großbuchstabe auf den Keller gelegt ($a \rightsquigarrow A$, $b \rightsquigarrow B$, ...).
- Wenn \$ eingelesen wird, bleibt der Keller unverändert.
- Anschließend wird für jedes neu eingelesene Zeichen überprüft, ob der passende Großbuchstabe auf dem Keller liegt. Dieser wird dann entfernt.
- Falls irgendwann keine Übereinstimmung festgestellt wird, blockiert der Kellerautomat.
- Falls immer Übereinstimmung herrscht, wird schließlich auch das Zeichen # entfernt, und der Automat akzeptiert mit leerem Keller.

## Simulation



- Zu Beginn einer jeden Berechnung enthält der Keller genau das Kellerinitialisierungszeichen #.
- Der Keller ist nicht beschränkt und kann beliebig wachsen, es gibt also unendlich viele mögliche Kellerinhalte. Mit anderen Worten: Im Gegensatz zu endlichen Automaten haben Kellerautomaten unendlich viele interne „Zustände“.
- Die von uns betrachteten Kellerautomaten akzeptieren immer mit leerem Keller (in diesem Fall gibt es auch keine Übergangsmöglichkeiten mehr). (Später betrachten wir auch eine Variante von Kellerautomaten, die mit Endzustand akzeptieren.)

## Definition

Ein Kellerautomat $M$ ist ein 6-Tupel $M = (Z, \Sigma, \Gamma, z_0, \delta, \#)$, wobei

- $Z$ die endliche Menge der Zustände,
- $\Sigma$ das Eingabealphabet,
- $\Gamma$ das Kelleralphabet,
- $z_0 \in Z$ der Startzustand,
- $\delta\colon Z \times \left(\Sigma \cup \{\varepsilon\}\right) \times \Gamma \to \mathcal{P}_e(Z \times \Gamma^*)$ die Überführungsfunktion und
- $\# \in \Gamma$ das Kellerinitialisierungszeichen ist.

**Bemerkung**: $\mathcal{P}_e(Z \times \Gamma^*)$ bezeichnet die Menge aller endlichen Teilmengen von $Z \times \Gamma^*$.

**Abkürzung:** PDA (pushdown automaton) oder NPDA (nondeterministic pushdown automaton).

Wir betrachten die Überführungsfunktion

$$\delta\colon Z \times \big(\Sigma \cup \{\varepsilon\}\big) \times \Gamma \to \mathcal{P}_e(Z \times \Gamma^*)\,.$$

$(z', B_1 \cdots B_k) \in \delta(z, a, A)$ bedeutet:

- wenn im Zustand $z$ das Eingabesymbol $a$ gelesen wird und das Zeichen $A$ als oberstes auf dem Keller liegt, dann
- wird $A$ vom Keller entfernt und durch $B_1 \cdots B_k$ ersetzt ($B_1$ liegt zuoberst) und der Automat geht in den Zustand $z'$ über.

Es kann auch $a = \varepsilon$ gelten. In diesem Fall wird kein Eingabesymbol eingelesen. Ebenso kann $k = 0$ gelten. In diesem Fall verkleinert sich der Kellerinhalt.

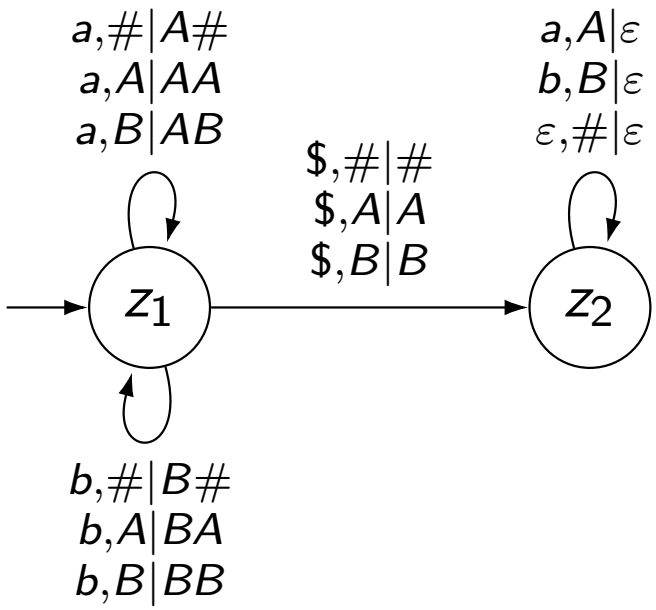
**Beispiel (vgl. Folie 11.2):**

PDA für $L = \left\{ a_1 a_2 \cdots a_n \$ a_n \cdots a_2 a_1 \;\middle|\; n \geq 0, a_1, \ldots, a_n \in \{a, b\} \right\}$:

$$M = \left( \{z_1, z_2\}, \{a, b, \$\}, \{\#, A, B\}, z_1, \delta, \# \right),$$

wobei $\delta$ folgendermaßen definiert ist
(wir schreiben $(z, a, A) \to (z', x)$, falls $(z', x) \in \delta(z, a, A)$):

$$\begin{array}{lll}
(z_1, a, \#) \to (z_1, A\#) & (z_1, a, A) \to (z_1, AA) & (z_1, a, B) \to (z_1, AB) \\
(z_1, b, \#) \to (z_1, B\#) & (z_1, b, A) \to (z_1, BA) & (z_1, b, B) \to (z_1, BB) \\
(z_1, \$, \#) \to (z_2, \#) & (z_1, \$, A) \to (z_2, A) & (z_1, \$, B) \to (z_2, B) \\
(z_2, a, A) \to (z_2, \varepsilon) & (z_2, b, B) \to (z_2, \varepsilon) & (z_2, \varepsilon, \#) \to (z_2, \varepsilon)
\end{array}$$


a,#|A#
a,A|AA
a,B|AB
$,#|#
$,A|A
$,B|B
a,A|ε
b,B|ε
ε,#|ε
z1
z2
b,#|B#
b,A|BA
b,B|BB

## Definition

Eine Konfiguration eines PDA ist ein Tripel $k \in Z \times \Sigma^* \times \Gamma^*$.

Bedeutung der Komponenten von $k = (z, w, \gamma) \in Z \times \Sigma^* \times \Gamma^*$:

- $z \in Z$ ist der aktuelle Zustand des PDA.
- $w \in \Sigma^*$ ist der noch zu lesende Teil der Eingabe.
- $\gamma \in \Gamma^*$ ist der aktuelle Kellerinhalt. Dabei steht das oberste Kellerzeichen ganz links.

Übergänge zwischen Konfigurationen ergeben sich aus der Überführungsfunktion $\delta$:

## Definition

Seien $\gamma \in \Gamma^*$, $A, B_1, \ldots, B_k \in \Gamma$, $w, w' \in \Sigma^*$ und $z, z' \in Z$. Dann gilt

$$(z, w, A\gamma) \vdash (z', w', B_1 \ldots B_k \gamma)$$

genau dann, wenn es $a \in \Sigma \cup \{\varepsilon\}$ gibt mit

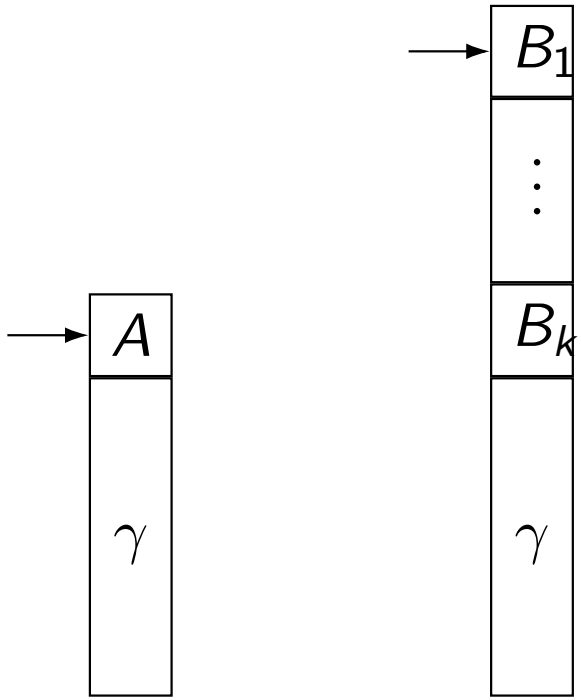
- $w = aw'$ und
- $(z', B_1 \ldots B_k) \in \delta(z, a, A)$

Gilt $a \in \Sigma$, so wird ein Zeichen der Eingabe gelesen. Falls $a = \varepsilon$ gilt, so nicht (d.h. es gilt $w = w'$).

Wir betrachten verschiedene Fälle von Werten der Überführungsfunktion $\delta$:

$(z', B_1 \cdots B_k) \in \delta(z, a, A)$

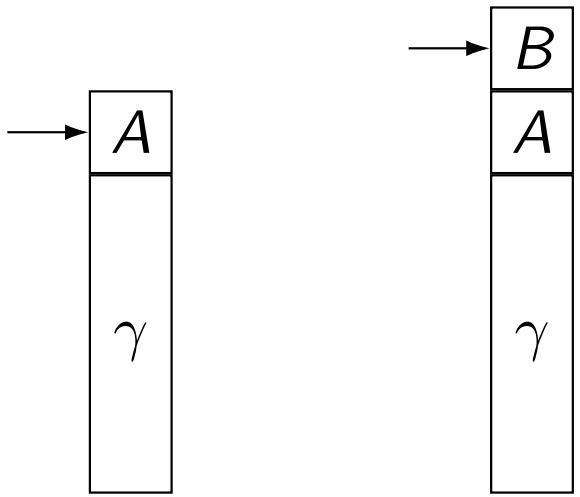
- Zeichen $a$ wird gelesen.
- Zustand ändert sich von $z$ nach $z'$.
- Symbol $A$ wird durch mehrere neue Symbole ersetzt:



Wir betrachten verschiedene Fälle von Werten der Überführungsfunktion $\delta$:

$(z', BA) \in \delta(z, a, A)$

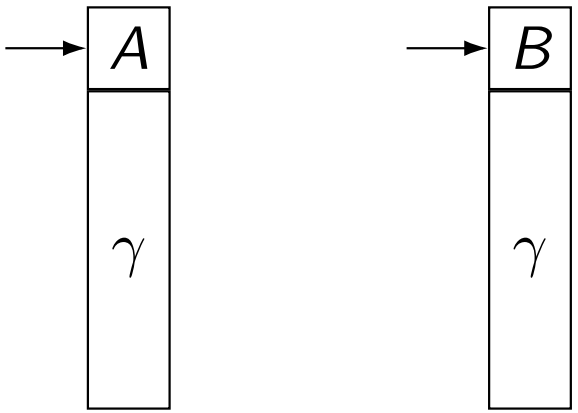
- Zeichen $a$ wird gelesen.
- Zustand ändert sich von $z$ nach $z'$.
- Symbol $B$ wird zusätzlich auf den Keller gelegt:



Wir betrachten verschiedene Fälle von Werten der Überführungsfunktion $\delta$:

$(z', B) \in \delta(z, a, A)$

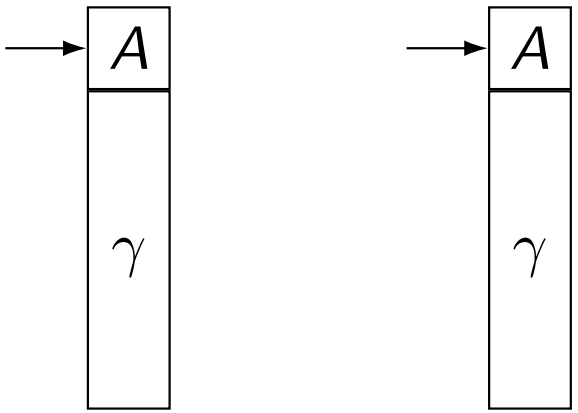
- Zeichen $a$ wird gelesen.
- Zustand ändert sich von $z$ nach $z'$.
- Symbol $A$ auf dem Keller wird durch $B$ ersetzt:



Wir betrachten verschiedene Fälle von Werten der Überführungsfunktion $\delta$:

$(z', A) \in \delta(z, a, A)$

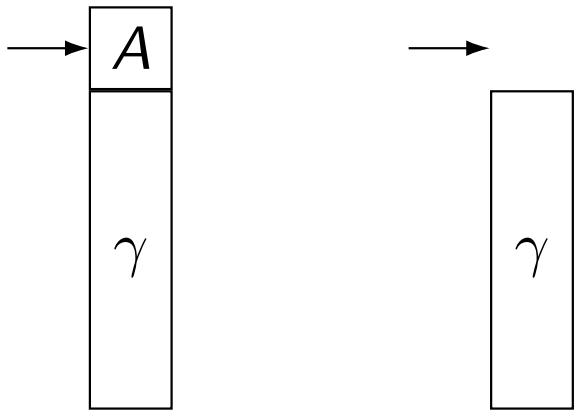
- Zeichen $a$ wird gelesen.
- Zustand ändert sich von $z$ nach $z'$.
- Keller bleibt unverändert:



Wir betrachten verschiedene Fälle von Werten der Überführungsfunktion $\delta$:

$(z', \varepsilon) \in \delta(z, a, A)$

- Zeichen $a$ wird gelesen.
- Zustand ändert sich von $z$ nach $z'$.
- Symbol $A$ wird vom Keller gelöscht:



Wir definieren $\vdash^*$ als die reflexive und transitive Hülle von $\vdash$.

**Definition**

Sei $M = (Z, \Sigma, \Gamma, z_0, \delta, \#)$ ein PDA. Dann ist die von $M$ akzeptierte Sprache:

$$L(M) = \left\{ x \in \Sigma^* \;\middle|\; \text{es gibt } z \in Z \text{ mit } (z_0, x, \#) \vdash^* (z, \varepsilon, \varepsilon) \right\}.$$

Das heißt, daß die akzeptierte Sprache diejenigen Wörter enthält, mit deren Hilfe es möglich ist, den Keller vollständig zu leeren.

Da Kellerautomaten jedoch nicht-deterministisch sind, kann es auch Berechnungen für dieses Wort geben, die den Keller nicht leeren.

**Beispiel (vgl. Folie 11.12):** ein PDA für die Sprache

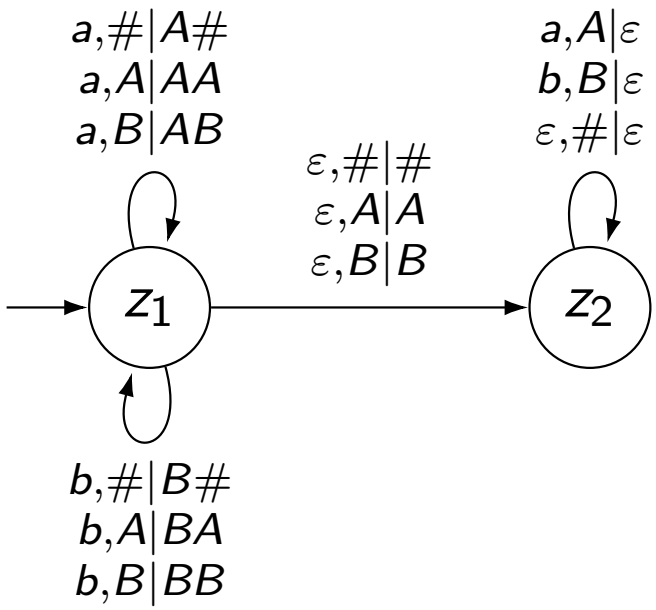
$$L = \left\{ a_1 a_2 \cdots a_n a_n \cdots a_2 a_1 \;\middle|\; n \geq 0, a_1, \ldots, a_n \in \{a, b\} \right\}.$$

(dies ist die Menge der „Palindrome gerader Länge“)

**Idee:** anstatt auf das Zeichen \$ zu warten, kann sich der Automat jederzeit nicht-deterministisch entscheiden, in den Zustand $z_2$ (= Keller abbauen) überzugehen.

Veränderte Überführungsfunktion $\delta$ (3. Zeile ist geändert):

$$\begin{array}{lll}
(z_1, a, \#) \to (z_1, A\#) & (z_1, a, A) \to (z_1, AA) & (z_1, a, B) \to (z_1, AB) \\
(z_1, b, \#) \to (z_1, B\#) & (z_1, b, A) \to (z_1, BA) & (z_1, b, B) \to (z_1, BB) \\
(z_1, \varepsilon, \#) \to (z_2, \#) & (z_1, \varepsilon, A) \to (z_2, A) & (z_1, \varepsilon, B) \to (z_2, B) \\
(z_2, a, A) \to (z_2, \varepsilon) & (z_2, b, B) \to (z_2, \varepsilon) & (z_2, \varepsilon, \#) \to (z_2, \varepsilon)
\end{array}$$


a,#|A#
a,A|AA
a,B|AB
ε,#|#
ε,A|A
ε,B|B
a,A|ε
b,B|ε
ε,#|ε
z1
z2
b,#|B#
b,A|BA
b,B|BB

Zum Beispiel ergeben sich diese Folgen von Konfigurationen für das Eingabewort *abba*:

- $(z_1, abba, \#) \vdash (z_1, bba, A\#) \vdash (z_2, bba, A\#)$
- $(z_1, abba, \#) \vdash (z_1, bba, A\#) \vdash (z_1, ba, BA\#) \vdash (z_2, ba, BA\#) \vdash (z_2, a, A\#) \vdash (z_2, \varepsilon, \#) \vdash (z_2, \varepsilon, \varepsilon)$
  Das Wort *abba* wird also akzeptiert.

Dieser Kellerautomat ist also nicht-deterministisch (d.h. eine Konfiguration kann mehrere Nachfolgerkonfigurationen haben).

# Die Greibach-Normalform

Wir wollen als nächstes zeigen, daß jede kontextfreie Sprache von einem PDA akzeptiert werden kann. Hierzu wandeln wir die kontextfreie Grammatik zunächst in eine Grammatik in Greibach-Normalform um.

## Definition

Eine kontextfreie Grammatik $G = (V, \Sigma, P, S)$ ist in Greibach-Normalform, falls alle Produktionen aus $P$ folgende Form haben:

$$A \to aB_1B_2 \dots B_k$$

mit $k \in \mathbb{N}$, $A, B_1, \cdots, B_k \in V$ und $a \in \Sigma$.

Die Greibach-Normalform garantiert, daß bei jedem Ableitungsschritt genau ein Alphabetsymbol entsteht.

## Satz

Aus einer kontextfreien Grammatik $G = (V, \Sigma, P, S)$ kann eine kontextfreie Grammatik $G'$ in Greibach-Normalform berechnet werden mit $L(G') = L(G) \setminus \{\varepsilon\}$.

**Beweis:** o.E. können wir annehmen, daß $G$ in Chomsky-Normalform ist.

Durch Streichen der Regel $(S \to \varepsilon)$ (falls sie existiert) ergibt sich eine kontextfreie Grammatik in Chomsky-Normalform für $L(G) \setminus \{\varepsilon\}$.

Wir gehen im folgenden davon aus,

- daß $G$ in Chomsky-Normalform ist und
- daß $(S \to \varepsilon)$ keine Regel ist, daß also $\varepsilon \notin L(G)$ gilt.

**Vorüberlegung:**

Angenommen in $P$ gibt es für ein Nichtterminal $A$ die Produktionen

$$A \to A\alpha_1 \mid \cdots \mid A\alpha_k \mid \beta_1 \mid \cdots \mid \beta_\ell$$

mit $\alpha_1, \ldots, \alpha_k, \beta_1, \ldots, \beta_\ell \in (V \cup \Sigma)^*$ und $\beta_1, \ldots, \beta_\ell$ beginnen nicht mit $A$.

Mit Hilfe von Linksableitungen lassen sich aus $A$ mittels dieser Regeln die Wörter aus

$$\{\beta_1, \ldots, \beta_\ell\} \{\alpha_1, \ldots, \alpha_k\}^*$$

erzeugen.

Betrachte die folgenden Produktionen (mit neuem Nichtterminal $B$):

$$A \to \beta_1 \mid \cdots \mid \beta_\ell \mid \beta_1 B \mid \cdots \mid \beta_\ell B$$
$$B \to \alpha_1 \mid \cdots \mid \alpha_k \mid \alpha_1 B \mid \cdots \mid \alpha_k B$$

Mit Hilfe von Rechtsableitungen lassen sich aus $A$ mittels dieser Regeln die Wörter aus

$$\{\beta_1, \ldots, \beta_\ell\} \{\alpha_1, \ldots, \alpha_k\}^*$$

erzeugen.

Ersetzt man also die ersten Regeln durch die zweiten, so ändert sich die Sprache der Grammatik nicht.

Sei nun $A_1, \ldots, A_m$ eine beliebige Aufzählung aller Nichtterminale von $G$.

Wir setzen

$$\mathit{first}(A_i) = \{X \in \Sigma \cup V \mid \exists \beta \in V^* \colon A_i \to X\beta\}\,,$$

d.h. dies ist die Menge der ersten Zeichen von rechten Seiten zu $A_i$.

**Schritt 1:** Mit dem Algorithmus auf der nächsten Folie formen wir $G$ in eine äquivalente kontextfreie Grammatik um, in der für alle Nichtterminale $A_i$ gilt

$$\mathit{first}(A_i) \subseteq \Sigma \cup \{A_{i+1}, A_{i+2}, \ldots, A_m\}\,.$$

**for** $i := 1$ **to** $m$ **do**
  **for** $j := 1$ **to** $i - 1$ **do**
    **for all** $(A_i \to A_j\alpha) \in P$ **do**
      Seien $A_j \to \beta_1 \mid \cdots \mid \beta_n$ alle Regeln mit linker Seite $A_j$.
      $P := \big(P \cup \{A_i \to \beta_1\alpha \mid \cdots \mid \beta_n\alpha\}\big) \setminus \{A_i \to A_j\alpha\}$
    **endfor**
  **endfor**
  $\{$assert: $\mathit{first}(A_k) \subseteq \Sigma \cup \{A_{k+1}, A_{k+2} \ldots, A_m\}$ f.a. $1 \leq k < i$
    $\mathit{first}(A_i) \subseteq \Sigma \cup \{A_i, A_{i+1} \ldots, A_m\}\}$
  **if** es gibt Produktionen der Form $A_i \to A_i\alpha$ **then**
    Wende die Transformation aus der Vorüberlegung auf $A_i$ an
    (dabei wird ein neues Nichtterminal $B_i$ eingeführt).
  **endif**
  $\{$assert: $\mathit{first}(A_k) \subseteq \Sigma \cup \{A_{k+1}, A_{k+2} \ldots, A_m\}$
    $\mathit{first}(B_k) \subseteq \Sigma \cup \{A_1, A_2, \ldots, A_m\}$ f.a. $1 \leq k \leq i\}$
**endfor**

Nach Schritt 1 gilt insbesondere $\mathit{first}(A_m) \subseteq \Sigma$.

**Schritt 2:** Der folgende Algorithmus erzwingt dies für alle $A_i$:

**for** $i := m - 1$ **downto** 1 **do**
    **forall** $(A_i \to A_j\alpha) \in P$ mit $j > i$ **do**
        Seien $A_j \to \beta_1 \mid \cdots \mid \beta_n$ alle Regeln mit linker Seite $A_j$.
        $P := \big(P \cup \{A_i \to \beta_1\alpha \mid \cdots \mid \beta_n\alpha\}\big) \setminus \{A_i \to A_j\alpha\}$
    **endfor**
**endfor**

Nach Schritt 2 sind alle Produktionen mit linker Seite $A_i$ $(1 \leq i \leq m)$ in Greibach-Normalform.

Aber: Die in Schritt 1 eingeführten Produktionen für die neuen Nichtterminale $B_i$ müssen nicht in Greibach-Normalform sein.

Sei $B_i \to A_j\alpha$ eine solche Regel, die die Greibach-Normalform verletzt.

Seien $A_j \to \beta_1 \mid \cdots \mid \beta_k$ alle Produktionen mit linker Seite $A_j$.

Dann beginnen $\beta_1, \ldots, \beta_k$ mit Terminalsymbolen.

Ersetze $B_i \to A_j\alpha$ durch $B_i \to \beta_1\alpha \mid \cdots \mid \beta_k\alpha$.

Nun ist die Grammatik in Greibach-Normalform. □

**Beispiel:** Sei $G$ die Grammatik in Chomsky-Normalform mit folgenden Produktionen:

$$\begin{aligned} A_1 &\to A_2 A_3 \\ A_2 &\to A_3 A_1 \mid b \\ A_3 &\to A_1 A_2 \mid a. \end{aligned}$$

In Schritt 1 wird nur die Produktion $A_3 \to A_1 A_2$ im Durchlauf $i = 3$ wie folgt ersetzt:

- Bei $j = 1$: $A_3 \to A_2 A_3 A_2$
- Bei $j = 2$: $A_3 \to A_3 A_1 A_3 A_2 \mid b A_3 A_2$

Nun wird ein neues Nichtterminal $B_3$ eingeführt, und die Produktionen

$$A_3 \to A_3 A_1 A_3 A_2 \mid b A_3 A_2 \mid a$$

werden ersetzt durch

$$\begin{aligned} A_3 &\to b A_3 A_2 B_3 \mid a B_3 \mid b A_3 A_2 \mid a \\ B_3 &\to A_1 A_3 A_2 B_3 \mid A_1 A_3 A_2. \end{aligned}$$

Wir haben also nach Schritt 1 folgende Grammatik vorliegen:

$$\begin{aligned} A_1 &\to A_2A_3 \\ A_2 &\to A_3A_1 \mid b \\ A_3 &\to bA_3A_2B_3 \mid aB_3 \mid bA_3A_2 \mid a \\ B_3 &\to A_1A_3A_2B_3 \mid A_1A_3A_2. \end{aligned}$$

Beachte: Rechte Seiten von Produktionen mit linker Seite $A_i$ beginnen mit Terminal oder mit $A_j$ für ein $j > i$.

Nach Schritt 2, Durchlauf $i = 2$:

$$\begin{aligned} A_1 &\to A_2A_3 \\ A_2 &\to bA_3A_2B_3A_1 \mid aB_3A_1 \mid bA_3A_2A_1 \mid aA_1 \mid b \\ A_3 &\to bA_3A_2B_3 \mid aB_3 \mid bA_3A_2 \mid a \\ B_3 &\to A_1A_3A_2B_3 \mid A_1A_3A_2 \end{aligned}$$

Nach Schritt 2, Durchlauf $i = 1$:

$$
\begin{aligned}
A_1 &\to bA_3A_2B_3A_1A_3 \mid aB_3A_1A_3 \mid bA_3A_2A_1A_3 \mid aA_1A_3 \mid bA_3 \\
A_2 &\to bA_3A_2B_3A_1 \mid aB_3A_1 \mid bA_3A_2A_1 \mid aA_1 \mid b \\
A_3 &\to bA_3A_2B_3 \mid aB_3 \mid bA_3A_2 \mid a \\
B_3 &\to A_1A_3A_2B_3 \mid A_1A_3A_2
\end{aligned}
$$

Nun muß noch in den rechten Seiten der $B_3$-Produktionen $A_1$ durch die rechten Seiten von $A_1$ ersetzt werden:

$$
\begin{aligned}
A_1 &\to bA_3A_2B_3A_1A_3 \mid aB_3A_1A_3 \mid bA_3A_2A_1A_3 \mid aA_1A_3 \mid bA_3 \\
A_2 &\to bA_3A_2B_3 \mid aB_3 \mid bA_3A_2 \mid aA_1 \mid b \\
A_3 &\to bA_3A_2B_3 \mid aB_3 \mid bA_3A_2 \mid a \\
B_3 &\to bA_3A_2B_3A_1A_3A_3A_2B_3 \mid aB_3A_1A_3A_3A_2B_3 \mid bA_3A_2A_1A_3A_3A_2B_3 \mid \\
&\quad aA_1A_3A_3A_2B_3 \mid bA_3A_3A_2B_3 \mid bA_3A_2B_3A_1A_3A_3A_2 \mid \\
&\quad aB_3A_1A_3A_3A_2 \mid bA_3A_2A_1A_3A_3A_2 \mid aA_1A_3A_3A_2 \mid bA_3A_3A_2
\end{aligned}
$$

# Zusammenfassung 11. Vorlesung

## in dieser Vorlesung neu

- PDAs (Hoffnung: sie akzeptieren genau die kontextfreien Sprachen)
- jede $\varepsilon$-freie kontextfreie Sprache kann durch eine Grammatik in Greibach-Normalform erzeugt werden

## kommende Vorlesung

- PDAs akzeptieren genau die kontextfreien Sprachen:
  - Konstruktion eines PDA aus einer Grammatik in Greibach-Normalform
  - Konstruktion einer kontextfreien Grammatik aus einem PDA („Tripelkonstruktion“)