LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

Formale Sprachen und Komplexität
Theoretische Informatik für Studierende der Medieninformatik
Sommersemester 2024

6c
Kellerautomaten

Prof. Dr. Jasmin Blanchette

Lehr- und Forschungseinheit für
Theoretische Informatik und Theorembeweisen

Stand: 29. Mai 2024
Basierend auf Folien von PD Dr. David Sabel

Überblick über Grammatiken und Maschinenmodelle

alle Sprachen

rekursiv aufzählbar	Typ 0-Grammatik $\ell \to r$ beliebig	Turingmaschine (det. & nichtdet.)
kontextsensitiv	Typ 1-Grammatik $\ell \to r$: $\lvert\ell\rvert \leq \lvert r\rvert$	linear beschränkte Turingmaschine (nichtdet.)
kontextfrei	Typ 2-Grammatik $A \to r$	Kellerautomat (nichtdet.)
regulär	Typ 3-Grammatik $A \to a[B]$	endlicher Automat (det. & nichtdet.)

Überblick über Grammatiken und Maschinenmodelle

alle Sprachen

rekursiv aufzählbar	Typ 0-Grammatik $\ell \to r$ beliebig	Turingmaschine (det. & nichtdet.)
kontextsensitiv	Typ 1-Grammatik $\ell \to r$: $\lvert\ell\rvert \leq \lvert r\rvert$	linear beschränkte Turingmaschine (nichtdet.)
kontextfrei	Typ 2-Grammatik $A \to r$	Kellerautomat (nichtdet.)
regulär	Typ 3-Grammatik $A \to a[B]$	endlicher Automat (det. & nichtdet.)

Begrenzungen der endlichen Automaten

- DFAs und NFAs haben fast keinen Speicher:
  Der einzige Speicher dort sind die Zustände, daher endlicher Speicher.

Begrenzungen der endlichen Automaten

- DFAs und NFAs haben fast keinen Speicher:
  Der einzige Speicher dort sind die Zustände, daher endlicher Speicher.
- Daher ist es z.B. unmöglich, $\{w\$\overline{w} \mid w \in \{a, b, c\}^*\}$ zu erkennen:
  Man müsste beim Lesen von w alle gelesenen Zeichen speichern,
  um sie dann beim Lesen von $\overline{w}$ (d.h. w rückwärts) zu vergleichen.

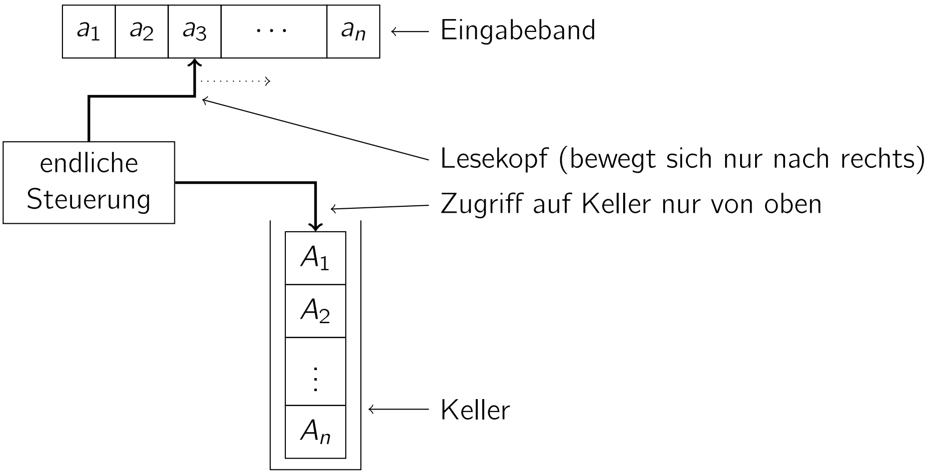
Kellerautomaten

Informelle Kurzfassung:

- Kellerautomaten fügen einen beliebig großen Kellerspeicher hinzu (Stack, LIFO-Speicher, last-in-first-out-Speicher).
- Der Keller ist ein Stapel, auf den nur von oben zugegriffen werden kann.
- Zustandsübergang:

	NFA mit ϵ-Übergängen	Kellerautomat
Eingabe	Zustand und Zeichen-oder-ε	Zustand, Zeichen-oder-ε und oberstes Symbol im Keller
Ausgabe	nächster Zustand	nächster Zustand und Sequenz von Kellersymbolen, die das erste Symbol ersetzen

Illustration eines Kellerautomaten



Definition eines Kellerautomaten

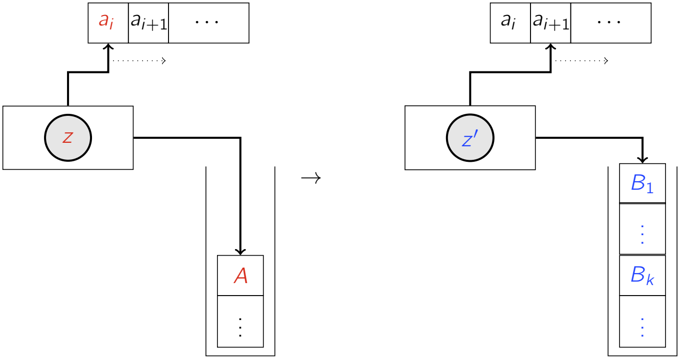
Definition

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

- Z ist eine endliche Menge von Zuständen
- Σ ist das (endliche) Eingabealphabet
- Γ ist das (endliche) Kelleralphabet
- $\delta : Z \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \mathcal{P}_e(Z \times \Gamma^*)$ ist die Überführungsfunktion
- $z_0 \in Z$ ist der Startzustand
- $\# \in \Gamma$ ist das Startsymbol im Keller.

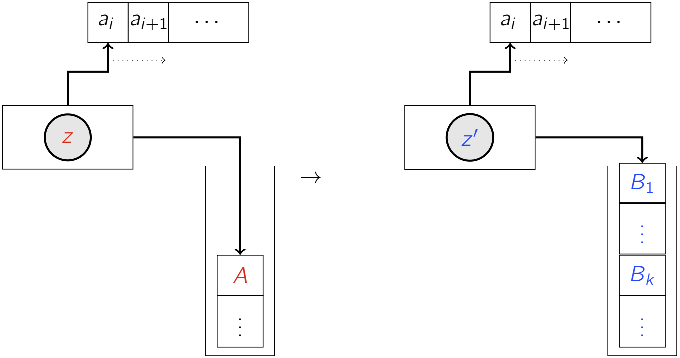
$\mathcal{P}_e(X)$ steht für die Menge der endlichen Teilmengen von X.

Illustration des Zustandsübergangs mit $a \in \Sigma$



$(z', B_1 \cdots B_k) \in \delta(z, a_i, A)$ bedeutet:
Im Zustand z bei Eingabe a_i und A oben auf dem Keller darf der PDA in z' wechseln.
Dabei wird A durch $B_1 \cdots B_k$ $(k \geq 0)$ ersetzt, wobei B_1 oben liegt.

Illustration des Zustandsübergangs mit ε



$(z', B_1 \cdots B_k) \in \delta(z, \varepsilon, A)$ bedeutet:
Im Zustand z bei A oben auf dem Keller darf der PDA in z' wechseln, ohne das der Lesekopf sich bewegt.
Dabei wird A durch $B_1 \cdots B_k$ $(k \geq 0)$ ersetzt, wobei B_1 oben liegt.

Bemerkungen

Mit unserer Definition:

- PDAs sind nichtdeterministisch.
- PDAs erlauben ε-Übergänge.
- PDAs haben keine Endzustände.

Bemerkungen

Mit unserer Definition:

- PDAs sind nichtdeterministisch.
- PDAs erlauben ε-Übergänge.
- PDAs haben keine Endzustände.

Zusätzlich:

- PDAs akzeptieren, wenn die Eingabe verarbeitet ist und der Keller leer ist.
- Am Anfang enthält der Keller $\#$.

Konfigurationen

- Während einer Berechnung mit dem PDA werden
  der aktuelle Zustand, die Resteingabe und der aktuelle Kellerinhalt
  buchführt.
- Diese Informationen werden durch eine PDA-Konfiguration dargestellt.

Konfigurationen

- Während einer Berechnung mit dem PDA werden der aktuelle Zustand, die Resteingabe und der aktuelle Kellerinhalt buchführt.
- Diese Informationen werden durch eine PDA-Konfiguration dargestellt.

Definition

Sei $M = (Z, \Sigma, \Gamma, \delta, z_0, \#)$ ein PDA.
Eine Konfiguration von M ist ein Tripel $(z, w, W) \in Z \times \Sigma^* \times \Gamma^*$, wobei:

- z ist der aktuelle Zustand
- w ist die Resteingabe
- W ist der aktuelle Kellerinhalt.

Übergangsrelation eines PDAs

Definition

Sei $M = (Z, \Sigma, \Gamma, \delta, z_0, \#)$ ein PDA.
Die Relation $\vdash_M \subseteq (Z \times \Sigma^* \times \Gamma^*) \times (Z \times \Sigma^* \times \Gamma^*)$ ist definiert durch

- $(z, a_1 \cdots a_n, A_1 \cdots A_m) \vdash_M (z', a_2 \cdots a_n, W A_2 \cdots A_m)$ falls $(z', W) \in \delta(z, a_1, A_1)$
- $(z, a_1 \cdots a_n, A_1 \cdots A_m) \vdash_M (z', a_1 \cdots a_n, W A_2 \cdots A_m)$ falls $(z', W) \in \delta(z, \varepsilon, A_1)$.

Übergangsrelation eines PDAs

Definition

Sei $M = (Z, \Sigma, \Gamma, \delta, z_0, \#)$ ein PDA.
Die Relation $\vdash_M \subseteq (Z \times \Sigma^* \times \Gamma^*) \times (Z \times \Sigma^* \times \Gamma^*)$ ist definiert durch

- $(z, a_1 \cdots a_n, A_1 \cdots A_m) \vdash_M (z', a_2 \cdots a_n, W A_2 \cdots A_m)$ falls $(z', W) \in \delta(z, a_1, A_1)$
- $(z, a_1 \cdots a_n, A_1 \cdots A_m) \vdash_M (z', a_1 \cdots a_n, W A_2 \cdots A_m)$ falls $(z', W) \in \delta(z, \varepsilon, A_1)$.

Weitere Notationen:

- $\vdash_M^*$ ist die reflexiv-transitive Hülle von $\vdash_M$.
- $\vdash_M^i$ ist die i-fache Anwendung von $\vdash_M$.
- Wenn M eindeutig ist, schreiben wir $\vdash$ statt $\vdash_M$.

Akzeptierte Sprache eines PDAs

Definition

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

$$L(M) := \{w \in \Sigma^* \mid (z_0, w, \#) \vdash^* (z, \varepsilon, \varepsilon) \text{ für ein } z \in Z\}$$

Zustandsgraph eines PDA

- Die Darstellung als Zustandsgraph ist analog zu der der NFAs mit ε-Übergängen.
- Für $(z', B_1 \cdots B_k) \in \delta(z, a, A)$ (mit $a \in \Sigma \cup \{\varepsilon\}$) zeichnen wir

$$z \xrightarrow{(A, a) \,:\, B_1 \cdots B_k} z'.$$

- Beachte, dass das Startsymbol im Keller bekannt sein muss (üblicherweise #).

Beispiel für einen PDA

PDA $M = (\{z_0, z_1\}, \{a, b\}, \{B, \#\}, \delta, z_0, \#)$ mit

$$\begin{array}{ccc} \delta(z_0, a, \#) = \{(z_0, B\#)\} & \delta(z_0, b, B) = \{(z_1, \varepsilon)\} & \delta(z_0, \varepsilon, \#) = \{(z_0, \varepsilon)\} \\ \delta(z_0, a, B) = \{(z_0, BB)\} & \delta(z_1, b, B) = \{(z_1, \varepsilon)\} & \delta(z_1, \varepsilon, \#) = \{(z_1, \varepsilon)\} \end{array}$$

und $\delta(z_i, c, A) = \emptyset$ sonst (für $c \in \{a, b, \varepsilon\}$).

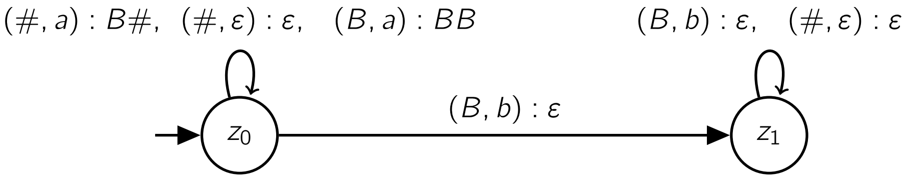
Beispiel für einen PDA

PDA $M = (\{z_0, z_1\}, \{a, b\}, \{B, \#\}, \delta, z_0, \#)$ mit

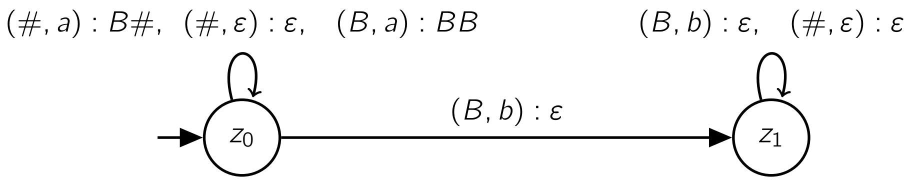
$$
\begin{array}{ccc}
\delta(z_0, a, \#) = \{(z_0, B\#)\} & \delta(z_0, b, B) = \{(z_1, \varepsilon)\} & \delta(z_0, \varepsilon, \#) = \{(z_0, \varepsilon)\} \\
\delta(z_0, a, B) = \{(z_0, BB)\} & \delta(z_1, b, B) = \{(z_1, \varepsilon)\} & \delta(z_1, \varepsilon, \#) = \{(z_1, \varepsilon)\}
\end{array}
$$

und $\delta(z_i, c, A) = \emptyset$ sonst (für $c \in \{a, b, \varepsilon\}$).

Zustandsgraph dazu:



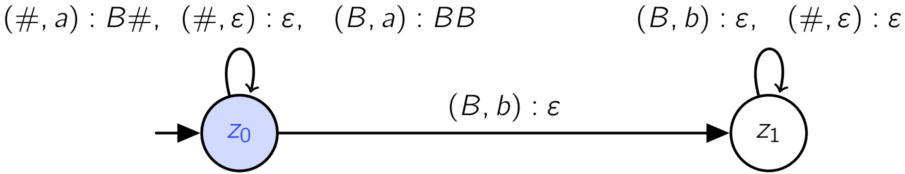
Beispiel für einen akzeptierenden Lauf



Eingabe: $aabb$

Keller: $\#$

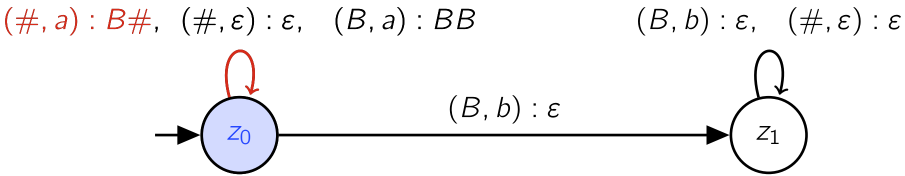
Beispiel für einen akzeptierenden Lauf



Eingabe: $aabb$

Keller: $\#$

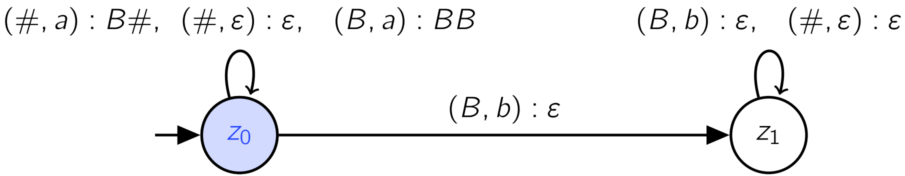
Beispiel für einen akzeptierenden Lauf



Eingabe: $aabb$

Keller: $\#$

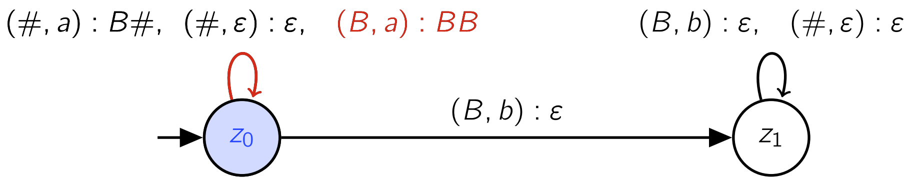
Beispiel für einen akzeptierenden Lauf



Eingabe: $aabb$

Keller: $B\#$

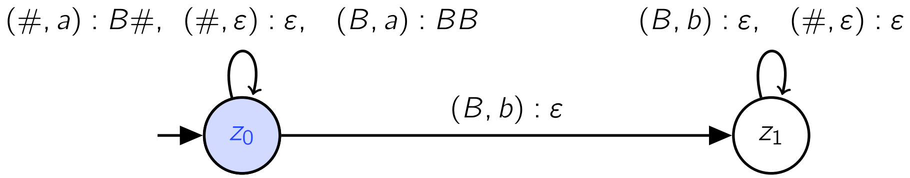
Beispiel für einen akzeptierenden Lauf



Eingabe: $aabb$

Keller: $B\#$

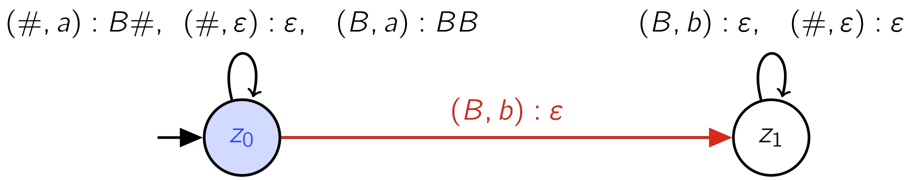
Beispiel für einen akzeptierenden Lauf



Eingabe: $aabb$

Keller: $BB\#$

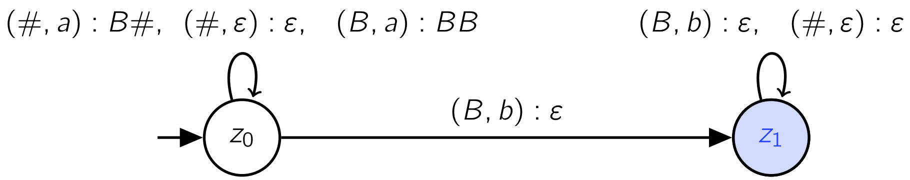
Beispiel für einen akzeptierenden Lauf



Eingabe: $aabb$

Keller: $BB\#$

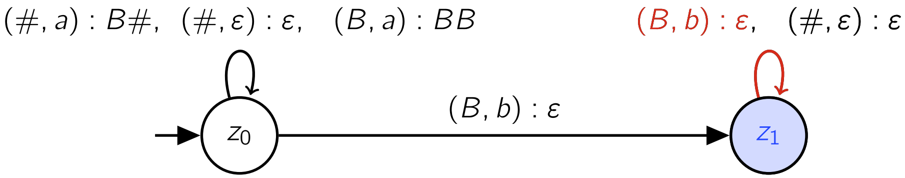
Beispiel für einen akzeptierenden Lauf



Eingabe: $aabb$

Keller: $B\#$

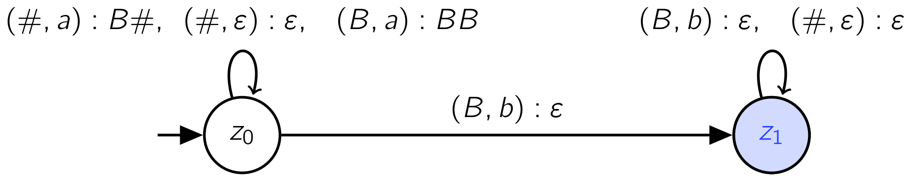
Beispiel für einen akzeptierenden Lauf



Eingabe: $aabb$

Keller: $B\#$

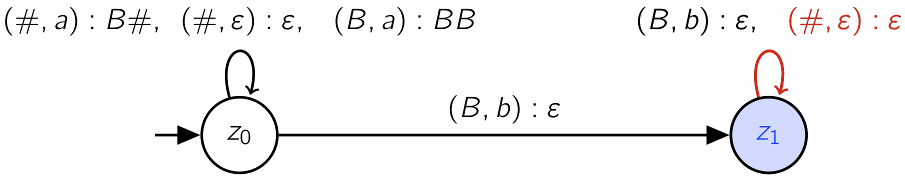
Beispiel für einen akzeptierenden Lauf



Eingabe: $aabb$

Keller: $\#$

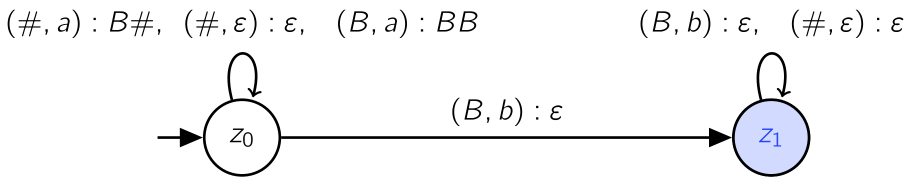
Beispiel für einen akzeptierenden Lauf



Eingabe: $aabb$

Keller: $\#$

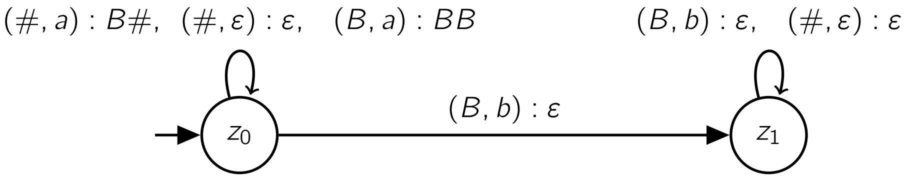
Beispiel für einen akzeptierenden Lauf



Eingabe: $aabb$

Keller: ε

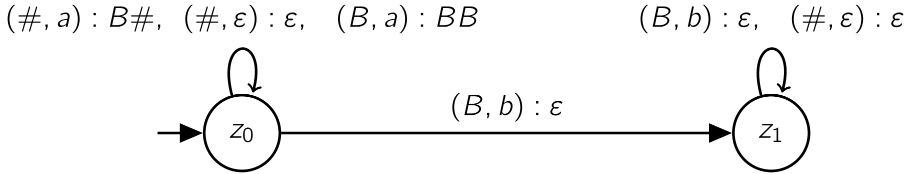
Beispiel für einen akzeptierenden Lauf



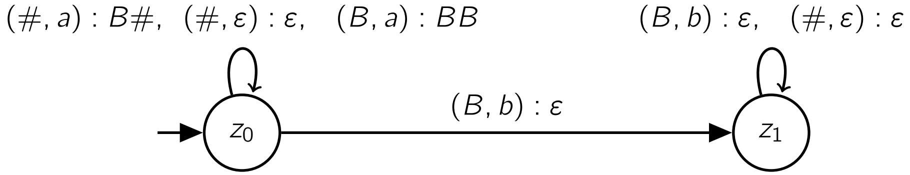
Eingabe: $aabb \in L(M)$

Keller: ε

Akzeptierte Sprache des Beispiels

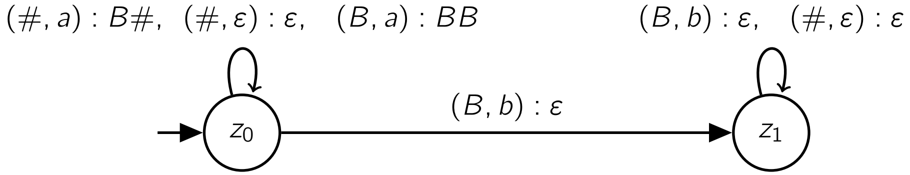


Akzeptierte Sprache des Beispiels



$L(M) = \{a^i b^i \mid i \in \mathbb{N}\}$ denn:

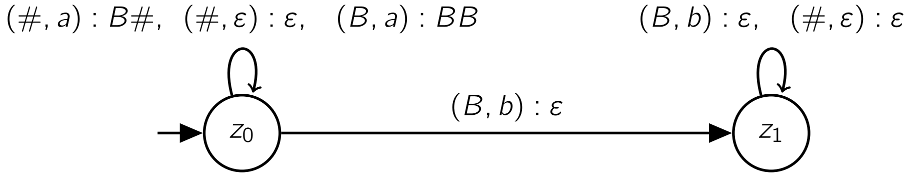
Akzeptierte Sprache des Beispiels



$L(M) = \{a^i b^i \mid i \in \mathbb{N}\}$ denn:

- M akzeptiert das Wort ε, denn $(z_0, \varepsilon, \#) \vdash (z_0, \varepsilon, \varepsilon)$.

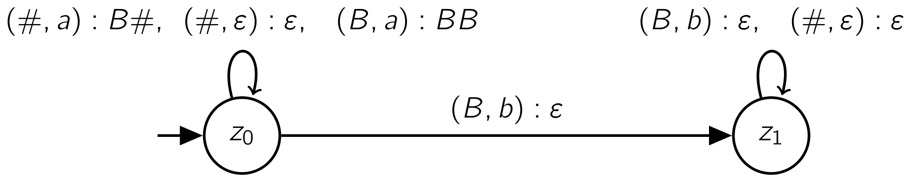
Akzeptierte Sprache des Beispiels



$L(M) = \{a^i b^i \mid i \in \mathbb{N}\}$ denn:

- M akzeptiert das Wort ε, denn $(z_0, \varepsilon, \#) \vdash (z_0, \varepsilon, \varepsilon)$.
- M akzeptiert das Wort $a^i b^i$ für $i > 0$, da $(z_0, a^i b^i, \#) \vdash (z_0, a^{i-1} b^i, B\#) \vdash^* (z_0, b^i, B^i\#) \vdash (z_1, b^{i-1}, B^{i-1}\#) \vdash^* (z_1, \varepsilon, \#) \vdash (z_1, \varepsilon, \varepsilon)$.

Akzeptierte Sprache des Beispiels



$L(M) = \{a^i b^i \mid i \in \mathbb{N}\}$ denn:

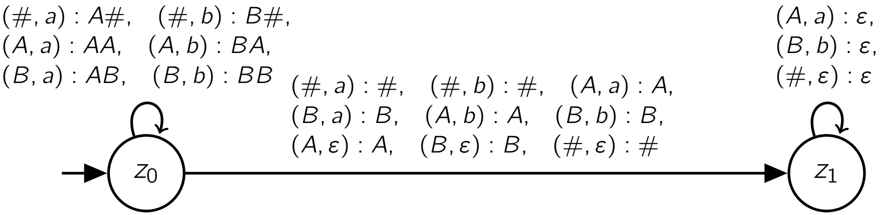
- M akzeptiert das Wort ε, denn $(z_0, \varepsilon, \#) \vdash (z_0, \varepsilon, \varepsilon)$.
- M akzeptiert das Wort $a^i b^i$ für $i > 0$, da $(z_0, a^i b^i, \#) \vdash (z_0, a^{i-1} b^i, B\#) \vdash^* (z_0, b^i, B^i\#) \vdash (z_1, b^{i-1}, B^{i-1}\#) \vdash^* (z_1, \varepsilon, \#) \vdash (z_1, \varepsilon, \varepsilon)$.
- Andere Wörter werden nicht akzeptiert.

Weiteres Beispiel für einen PDA

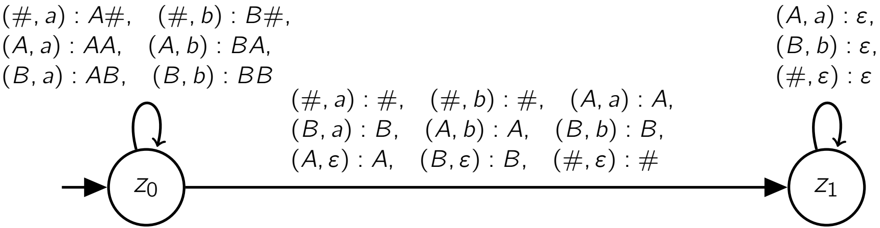
Sei $M = (\{z_0, z_1\}, \{a, b\}, \{A, B, \#\}, \delta, z_0, \#)$ mit

$$\begin{aligned}
\delta(z_0, a, \#) &= \{(z_0, A\#), (z_1, \#)\} & \qquad \delta(z_0, \varepsilon, A) &= \{(z_1, A)\} \\
\delta(z_0, b, \#) &= \{(z_0, B\#), (z_1, \#)\} & \delta(z_0, \varepsilon, B) &= \{(z_1, B)\} \\
\delta(z_0, a, A) &= \{(z_0, AA), (z_1, A)\} & \delta(z_0, \varepsilon, \#) &= \{(z_1, \#)\} \\
\delta(z_0, b, A) &= \{(z_0, BA), (z_1, A)\} & \delta(z_1, a, A) &= \{(z_1, \varepsilon)\} \\
\delta(z_0, a, B) &= \{(z_0, AB), (z_1, B)\} & \delta(z_1, b, B) &= \{(z_1, \varepsilon)\} \\
\delta(z_0, b, B) &= \{(z_0, BB), (z_1, B)\} & \delta(z_1, \varepsilon, \#) &= \{(z_1, \varepsilon)\}
\end{aligned}$$

und $\delta(z_i, c, C) = \emptyset$ sonst (für $c \in \{a, b, \varepsilon\}$). Zustandsgraph dazu:

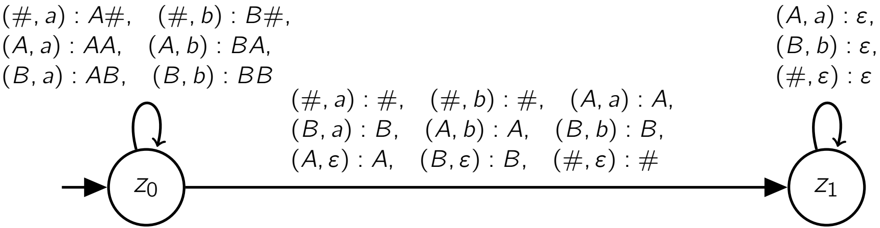


Akzeptierte Sprache des weiteren Beispiels



$L(M) = \{w \in \{a, b\}^* \mid w \text{ ist ein Palindrom}\}$ denn:

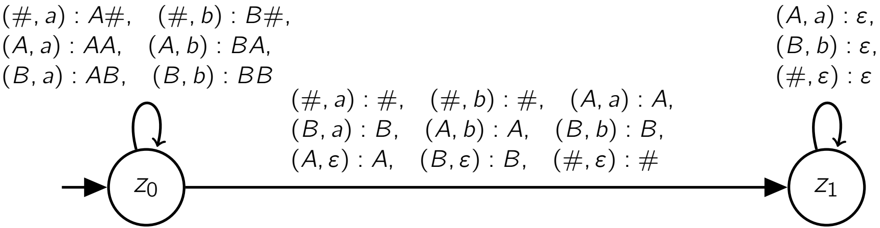
Akzeptierte Sprache des weiteren Beispiels



$L(M) = \{w \in \{a, b\}^* \mid w \text{ ist ein Palindrom}\}$ denn:

- In z_0 werden die gelesenen Zeichen (als A, B) auf den Keller gelegt.

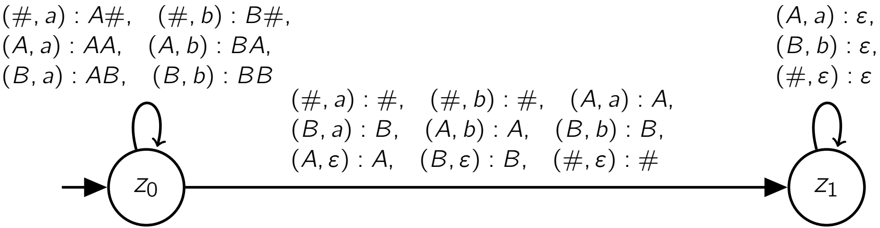
Akzeptierte Sprache des weiteren Beispiels



$L(M) = \{w \in \{a, b\}^* \mid w \text{ ist ein Palindrom}\}$ denn:

- In z_0 werden die gelesenen Zeichen (als A, B) auf den Keller gelegt.
- In z_1 werden sie dann wieder abgearbeitet (durch Lesen von a, b).

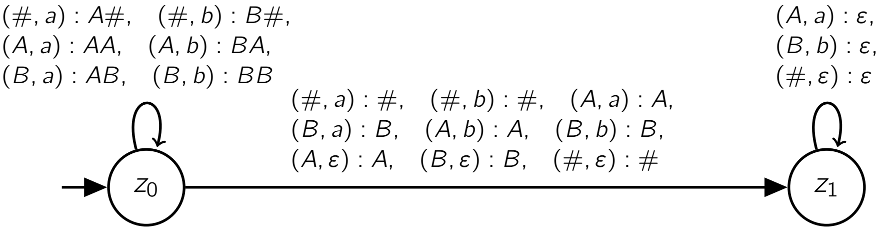
Akzeptierte Sprache des weiteren Beispiels



$L(M) = \{w \in \{a, b\}^* \mid w \text{ ist ein Palindrom}\}$ denn:

- In z_0 werden die gelesenen Zeichen (als A, B) auf den Keller gelegt.
- In z_1 werden sie dann wieder abgearbeitet (durch Lesen von a, b).
- Wechsel von z_0 zu z_1 mit einem Zeichen (für Palindrome $ua\overline{u}$, $ub\overline{u}$) oder mit ε (für Palindrome $u\overline{u}$).

Akzeptierte Sprache des weiteren Beispiels



$L(M) = \{w \in \{a, b\}^* \mid w$ ist ein Palindrom$\}$ denn:

- In z_0 werden die gelesenen Zeichen (als A, B) auf den Keller gelegt.
- In z_1 werden sie dann wieder abgearbeitet (durch Lesen von a, b).
- Wechsel von z_0 zu z_1 mit einem Zeichen (für Palindrome $ua\overline{u}$, $ub\overline{u}$) oder mit ε (für Palindrome $u\overline{u}$).
- Der Nichtdeterminismus entdeckt den richtigen Zeitpunkt des Wechsels.

PDAs mit Erzeugung von höchstens zwei Kellersymbolen

Lemma

Für jeden PDA $M = (Z, \Sigma, \Gamma, \delta, z_0, \#)$ gibt es einen PDA $M' = (Z, \Sigma, \Gamma', \delta', z_0, \#)$ mit $L(M') = L(M)$, sodass wenn $(z', B_1 \cdots B_k) \in \delta'(z, a, A)$ (für $a \in \Sigma \cup \{\varepsilon\}$), dann ist $k \leq 2$.

Beweis Transformiere M in M' wie folgt.

PDAs mit Erzeugung von höchstens zwei Kellersymbolen

Lemma

Für jeden PDA $M = (Z, \Sigma, \Gamma, \delta, z_0, \#)$ gibt es einen PDA $M' = (Z, \Sigma, \Gamma', \delta', z_0, \#)$ mit $L(M') = L(M)$, sodass wenn $(z', B_1 \cdots B_k) \in \delta'(z, a, A)$ (für $a \in \Sigma \cup \{\varepsilon\}$), dann ist $k \leq 2$.

Beweis Transformiere M in M' wie folgt.

Sei $(z', B_1 \cdots B_k) \in \delta(z, a, A)$ mit $A \in \Gamma$ und $a \in \Sigma \cup \{\varepsilon\}$.

PDAs mit Erzeugung von höchstens zwei Kellersymbolen

Lemma

Für jeden PDA $M = (Z, \Sigma, \Gamma, \delta, z_0, \#)$ gibt es einen PDA $M' = (Z, \Sigma, \Gamma', \delta', z_0, \#)$ mit $L(M') = L(M)$, sodass wenn $(z', B_1 \cdots B_k) \in \delta'(z, a, A)$ (für $a \in \Sigma \cup \{\varepsilon\}$), dann ist $k \leq 2$.

Beweis Transformiere M in M' wie folgt.

Sei $(z', B_1 \cdots B_k) \in \delta(z, a, A)$ mit $A \in \Gamma$ und $a \in \Sigma \cup \{\varepsilon\}$.

- Falls $k \leq 2$, dann $(z', B_1 \cdots B_k) \in \delta'(z, a, A)$.

PDAs mit Erzeugung von höchstens zwei Kellersymbolen

Lemma

Für jeden PDA $M = (Z, \Sigma, \Gamma, \delta, z_0, \#)$ gibt es einen PDA $M' = (Z, \Sigma, \Gamma', \delta', z_0, \#)$ mit $L(M') = L(M)$, sodass wenn $(z', B_1 \cdots B_k) \in \delta'(z, a, A)$ (für $a \in \Sigma \cup \{\varepsilon\}$), dann ist $k \leq 2$.

Beweis Transformiere M in M' wie folgt.

Sei $(z', B_1 \cdots B_k) \in \delta(z, a, A)$ mit $A \in \Gamma$ und $a \in \Sigma \cup \{\varepsilon\}$.

- Falls $k \leq 2$, dann $(z', B_1 \cdots B_k) \in \delta'(z, a, A)$.
- Falls $k > 2$, dann
  - $(z, C_k B_k) \in \delta'(z, a, A)$
  - $\delta'(z, \varepsilon, C_i) = \{(z, C_{i-1} B_{i-1})\}$ für jedes i mit $4 \leq i \leq k$
  - $\delta'(z, \varepsilon, C_3) = \{(z', B_1 B_2)\}$

  wobei $C_3, \ldots, C_k \in \Gamma'$ neue Kellersymbole sind.
  (Diese werden jeweils neu erzeugt pro ersetztem Eintrag.) $\square$

Akzeptanz durch Endzustände

Definition

Ein (nichtdeterministischer) Kellerautomat mit Endzuständen (PDA mit Endzuständen) ist ein 7-Tupel $M = (Z, \Sigma, \Gamma, \delta, z_0, \#, E)$, wobei:

- Z ist eine endliche Menge von Zuständen
- Σ ist das (endliche) Eingabealphabet
- Γ ist das (endliche) Kelleralphabet
- $\delta : Z \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \mathcal{P}_e(Z \times \Gamma^*)$ ist die Überführungsfunktion
- $z_0 \in Z$ ist der Startzustand
- $\# \in \Gamma$ ist das Startsymbol im Keller und
- $E \subseteq Z$ ist die Menge der Endzustände.

Akzeptanz durch Endzustände

Definition

Sei $M = (Z, \Sigma, \Gamma, \delta, z_0, \#, E)$ ein PDA mit Endzuständen.
Die von M akzeptierte Sprache ist

$$L(M) := \{w \in \Sigma^* \mid (z_0, w, \#) \vdash^* (z, \varepsilon, W) \text{ für } z \in E \text{ und } W \in \Gamma^*\}$$

Äquivalenz von PDAs mit und ohne Endzustände

Lemma

Für jeden PDA mit Endzuständen M kann ein PDA M' ohne Endzustände konstruiert werden, sodass $L(M') = L(M)$ gilt.

Lemma

Für jeden PDA M ohne Endzustände kann ein PDA mit Endzuständen M' konstruiert werden, sodass $L(M') = L(M)$ gilt.

Satz

PDAs mit Endzuständen und PDAs ohne Endzustände sind äquivalente Formalismen.

Beweis Siehe Skript. $\square$

Anwendungen von Kellerautomaten

- Sie dienen als Implementierung für Tools wie yacc, ANTLR und PLY, die syntaktische Analyser ("Parser") generieren.
- Sie werden für die Analyse von Netzwerken verwendet.
- Sie werden für die automatische Verifikation von (z.B. C++- oder Java-)Programmen, inklusive Prozeduren und Rekursion, verwendet.