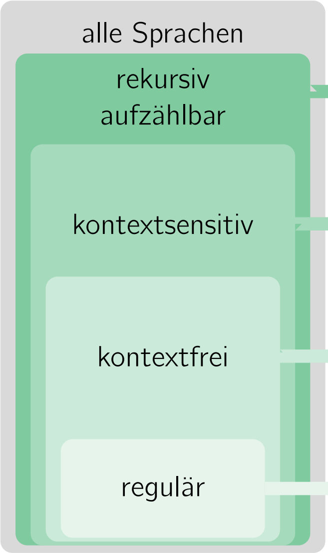
Kontextfreie Sprachen: Kellerautomaten

Prof. Dr. Jasmin Blanchette

Lehr- und Forschungseinheit für
Theoretische Informatik

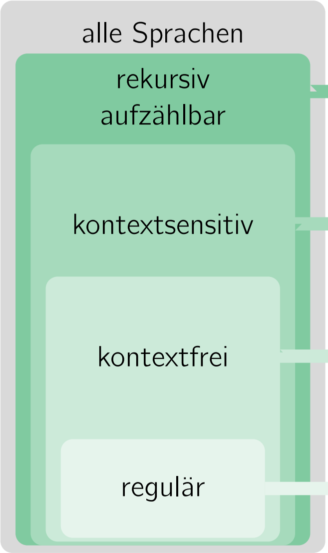
Stand: 30. Mai 2023
Folien ursprünglich von PD Dr. David Sabel

Überblick: Grammatiken & Maschinenmodelle für die Chomsky-Hierarchie



Grammatik		Maschinenmodell
Typ 0-Grammatik $\ell \to r$ bel.	$\longleftrightarrow$	Turingmaschine (det. & nichtdet.)
Typ 1-Grammatik $\ell \to r \Rightarrow \lvert\ell\rvert \leq \lvert r\rvert$	$\longleftrightarrow$	linear beschränkte Turingmaschine (nichtdet.)
Typ 2-Grammatik $A \to r$	$\longleftrightarrow$	Kellerautomat (nichtdet.)
Typ 3-Grammatik $A \to a[B]$	$\longleftrightarrow$	endlicher Automat (det. & nichtdet.)

Überblick: Grammatiken & Maschinenmodelle für die Chomsky-Hierarchie



Grammatik		Maschinenmodell
Typ 0-Grammatik $\ell \to r$ bel.	$\leftrightarrow$	Turingmaschine (det. & nichtdet.)
Typ 1-Grammatik $\ell \to r \Rightarrow \lvert\ell\rvert \leq \lvert r\rvert$	$\leftrightarrow$	linear beschränkte Turingmaschine (nichtdet.)
Typ 2-Grammatik $A \to r$	$\leftrightarrow$	Kellerautomat (nichtdet.)
Typ 3-Grammatik $A \to a[B]$	$\leftrightarrow$	endlicher Automat (det. & nichtdet.)

Hintergrund zu Kellerautomaten

- Endliche Automaten (DFA & NFA) haben fast **keinen Speicher**.
- Einziger Speicher dort sind die Zustände, daher endlicher Speicher.
- Daher 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 $\overline{w}$ zu vergleichen.

Kellerautomaten fügen einen beliebig großen Speicher hinzu.

Kellerspeicher

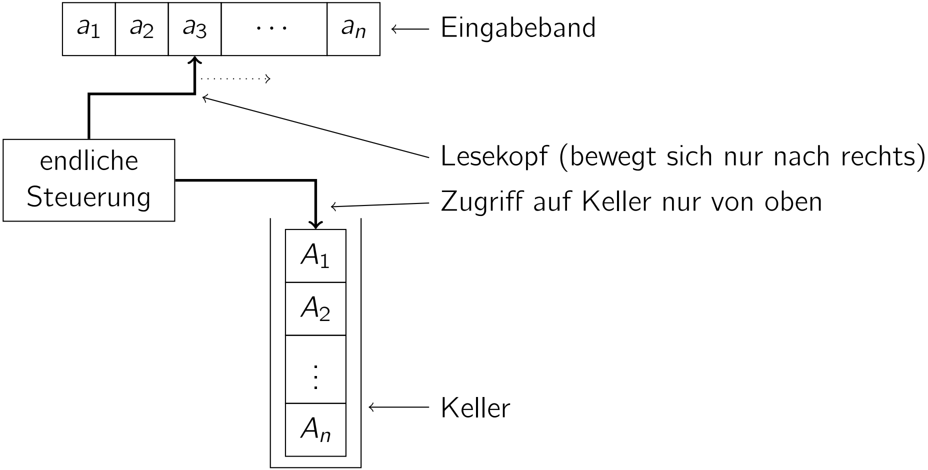
- Kellerautomaten haben Kellerspeicher
  (Stack, LIFO-Speicher, last-in-first-out-Speicher).

- Unendlich großer Speicher als Stapel,
  auf den **nur von oben** zugegriffen werden kann.

- Zustandsübergang:

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

Kellerautomat: Illustration



Kellerautomaten: Definition

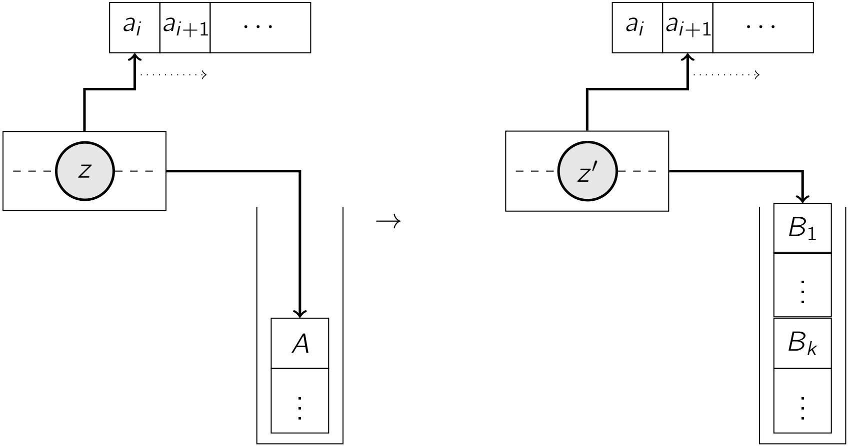
Definition (Kellerautomat, PDA)

Ein (nichtdeterministischer) Kellerautomat (PDA, pushdown automaton) ist ein 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 und
- $\# \in \Gamma$ ist das Startsymbol im Keller.

$(z', B_1 \cdots B_k) \in \delta(z, a, A)$ bedeutet: im Zustand z bei Eingabe a und A oben auf dem Keller darf der PDA in Zustand z' wechseln:
Dabei wird A durch $B_1 \cdots B_k$ ersetzt (B_1 liegt oben; $k = 0$ ist erlaubt).

Illustration: Zustandsübergang



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

Bemerkungen

Mit unserer Definition von PDAs:

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

Wir werden sehen:

- Akzeptieren: Wenn Eingabe verarbeitet und Keller leer
- Am Anfang: Keller enthält $\#$

Konfigurationen

- Buchführen während einer Berechnung mit dem PDA:
  aktueller Zustand, Resteingabe, aktueller Kellerinhalt
- Wird dargestellt durch PDA-Konfiguration

Definition (Konfiguration eines Kellerautomaten)

Sei $M = (Z, \Sigma, \Gamma, \delta, z_0, \#)$ ein PDA.
Eine Konfiguration von M ist ein Tripel (z, w, W)
mit $z \in Z$, $w \in \Sigma^*$, $W \in \Gamma^*$.
Die Menge aller Konfigurationen für M ist daher $Z \times \Sigma^* \times \Gamma^*$.

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

Übergangsrelation

Definition (Übergangsrelation $\vdash_M$ für PDA-Konfigurationen)

Für einen PDA $M = (Z, \Sigma, \Gamma, \delta, z_0, \#)$ definieren wir

$$\vdash_M \subseteq (Z \times \Sigma^* \times \Gamma^*) \times (Z \times \Sigma^* \times \Gamma^*)$$

durch

- $(z, a_1 \cdots a_n, A_1 \cdots A_m) \vdash_M (z', a_2 \cdots a_n, WA_2 \cdots A_m)$ falls $(z', W) \in \delta(z, a_1, A_1)$ und
- $(z, a_1 \cdots a_n, A_1 \cdots A_m) \vdash_M (z', a_1 \cdots a_n, WA_2 \cdots A_m)$ falls $(z', W) \in \delta(z, \varepsilon, A_1)$.

Weitere Notation:

- $\vdash_M^* =$ reflexiv-transitive Hülle von $\vdash_M$
- $\vdash_M^i =$ i-fache Anwendung von $\vdash_M$
- Wenn M eindeutig: $\vdash$ statt $\vdash_M$

PDA: Akzeptierte Sprache

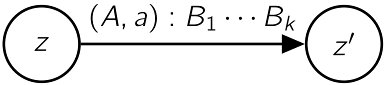
Definition (Akzeptierte Sprache eines PDA)

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

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

Notation als Zustandsgraph

- Darstellung analog zu DFA/NFA
- Für $(z', B_1 \cdots B_k) \in \delta(z, a, A)$ zeichnen wir



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

Beispiel (1)

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

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

und $\delta(z_i, c, A) = \emptyset$ in allen anderen Fällen.

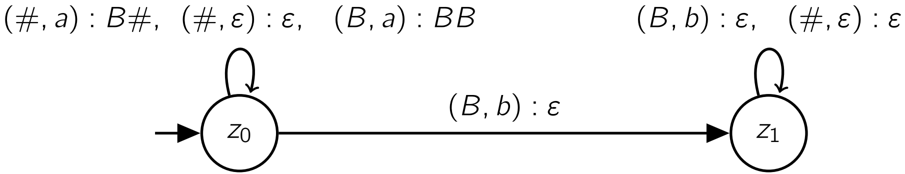
Beispiel (1)

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

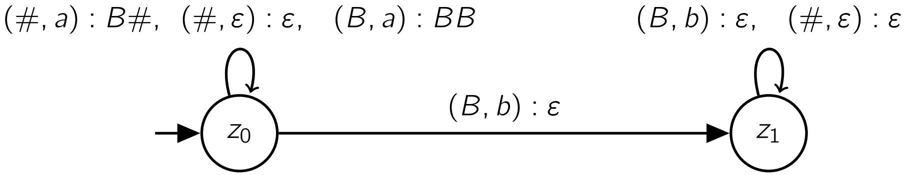
$$
\begin{array}{lll}
\delta(z_0, a, \#) = \{(z_0, B\#)\} & \delta(z_0, b, B) = \{(z_1, \varepsilon)\} & \delta(z_1, \varepsilon, \#) = \{(z_1, \varepsilon)\} \\
\delta(z_0, a, B) = \{(z_0, BB)\} & \delta(z_1, b, B) = \{(z_1, \varepsilon)\} & \delta(z_0, \varepsilon, \#) = \{(z_0, \varepsilon)\}
\end{array}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

und $\delta(z_i, c, A) = \emptyset$ in allen anderen Fällen.

Zustandsgraph dazu:



Beispiel (2)



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

$$(z_0, a^i b^i, \#) \vdash (z_0, a^{i-1} b^i, B\#) \vdash^* (z_0, b^i, B^i\#) \vdash (z_1, b^{i-1}, B^{i-1}\#) \vdash^* (z_1, \varepsilon, \#) \vdash (z_1, \varepsilon, \varepsilon)$$

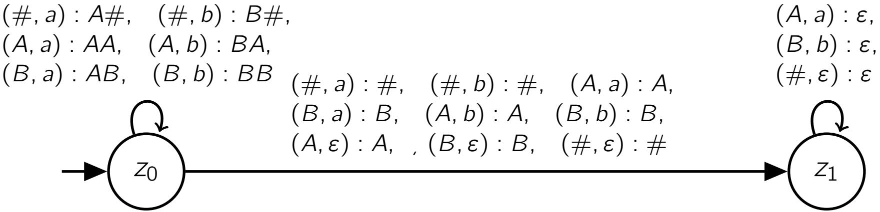
- andere Wörter werden nicht akzeptiert:
  - Für jedes gelesene a, gibt es B im Keller, das durch Lesen von b abgebaut werden muss.
  - Verbleiben mit a in z_0, Wechsel mit b in z_1: Dort können nur b's gelesen werden.
- $L(M) = \{a^i b^i \mid i \in \mathbb{N}\}$

Weiteres Beispiel

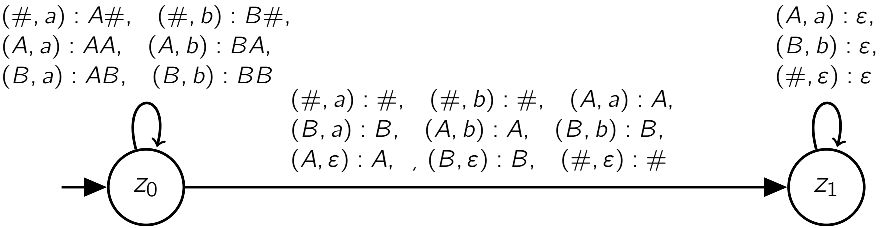
Sei $M = (\{z_0, z_1\}, \{a, b\}, \{A, B, \#\}, \delta, z_0, \#)$ mit

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

und $\delta(z_i, c, C) = \emptyset$ für alle anderen Fälle.



Weiteres Beispiel (2)



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

- 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 $u\overline{u}$).
- Richtiger Zeitpunkt des Wechsels: Macht der Nichtdeterminismus.

Akzeptanz durch Endzustände

Definition (PDA mit Endzuständen)

Ein (nichtdeterministischer) Kellerautomat mit Endzuständen (PDA mit Endzuständen) ist ein 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.

Ein PDA mit Endzuständen akzeptiert die Sprache

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

Äquivalenz: Akzeptanz durch Endzustände vs. durch leeren Keller

Lemma

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

Lemma

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

Satz

PDAs mit Endzuständen und PDAs ohne Endzustände (mit Akzeptanz durch leeren Keller) sind äquivalente Formalismen.

Beweise dazu sind im Skript.

Äquivalenz: PDAs und CFLs

Theorem

Kellerautomaten erkennen genau die kontextfreien Sprachen.

Wir erläutern hier nur die Idee wie man für eine CFG einen PDA erstellt, der dieselbe Sprache akzeptiert:
Für $G = (V, \Sigma, P, S)$ in Greibach-Normalform sei $M = (\{z_0\}, \Sigma, V, \delta, z_0, S)$ ein PDA, sodass $\delta(z_0, a, A) := \{(z_0, B_1 \cdots B_n) \mid (A \to aB_1 \cdots B_n) \in P\}$ und $\delta(z_0, \varepsilon, A) = \emptyset$.

- M simuliert Linksableitung $S \Rightarrow^* w$.
- Da G in Greibach-Normalform ist, sieht eine Linksableitung nach i Schritten immer so aus:

$$S \Rightarrow^i a_1 \cdots a_i B_1 \cdots B_j$$

- Start mit Eingabe w und S auf dem Keller.
- Nach i Schritten, ist $a_1 \cdots a_i$ verarbeitet und $B_1 \cdots B_j$ auf dem Keller.

Genauer Beweis und Richtung PDA $\to$ CFG: Nächste Stunde (nur FSK)