TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik, Institut für Künstliche Intelligenz, Professur für Computational Logic

FORMALE SYSTEME

16. Vorlesung: Kellerautomaten & CFGs

Hannes Straß

TU Dresden, 16. Dezember 2024

Ausblick

Plan

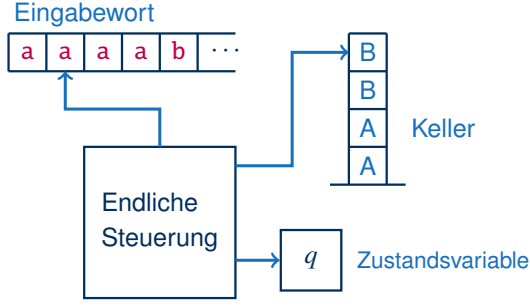
- Grammatiken
- Reguläre Sprachen
  - DFA und NFA
  - Umformungen, Abschlusseigenschaften, Minimalisierung
  - Pumping-Lemma
- Kontextfreie Sprachen
  - Wortproblem (CYK)
  - Pumping-Lemma
  - Kellerautomaten

Plan

- Grammatiken
- Reguläre Sprachen
  - DFA und NFA
  - Umformungen, Abschlusseigenschaften, Minimalisierung
  - Pumping-Lemma
- Kontextfreie Sprachen
  - Wortproblem (CYK)
  - Pumping-Lemma
  - Kellerautomaten
- Typ 1 und Typ 0
  - Turingmaschinen
  - (Un-)Entscheidbarkeit
- Aussagenlogik
  - Syntax und Semantik
  - Algorithmen zum logischen Schließen
  - Die Komplexitätsklasse NP

Rückblick

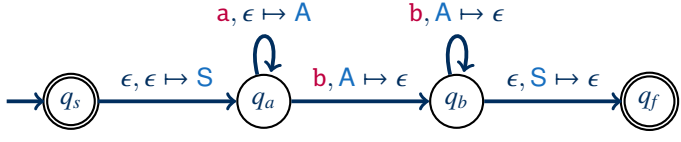
Kellerautomat = NFA + Stapelspeicher



Eine mögliche Konfiguration des PDA während der Erkennung ist gegeben durch den Zustand $q \in Q$, den Inhalt des Kellers $\gamma \in \Gamma^*$ und das noch zu lesende Restwort $w \in \Sigma^*$.

Kellerautomaten (PDAs)

Beispiel eines PDA für $\left\{ \mathrm{a}^i \mathrm{b}^i \;\middle|\; i \geq 0 \right\}$:



Ein Kellerautomat (international: „PDA“, „Pushdown Automaton“) $\mathcal{P}$ ist ein Tupel $\mathcal{P} = \langle Q, \Sigma, \Gamma, \delta, Q_0, F \rangle$ mit Zustandsmenge Q, Eingabealphabet Σ, Kelleralphabet Γ, Startzuständen Q_0, Endzuständen F und Übergangsfunktion δ:

$$Q \times \Sigma_\epsilon \times \Gamma_\epsilon \to 2^{Q \times \Gamma_\epsilon}$$

wobei $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$ und $\Gamma_\epsilon = \Gamma \cup \{\epsilon\}$.

Ausdrucksstärke von PDAs

Ausdrucksstärke von PDAs

Man kann nun formal untersuchen, welche Sprachen durch PDAs akzeptiert werden können. Wir erhalten das erhoffte Resultat:

Satz:
Eine Sprache ist genau dann kontextfrei, wenn sie von einem PDA akzeptiert wird.

Der Beweis erfolgt in zwei Schritten:

(1) Umwandlung Typ-2-Grammatik $\rightsquigarrow$ PDA

(2) Umwandlung PDA $\rightsquigarrow$ Typ-2-Grammatik

PDA $\rightsquigarrow$ Grammatik

Satz: Für jeden PDA $\mathcal{P}$ gibt es eine kontextfreie Grammatik $G_{\mathcal{P}}$, so dass $\mathbf{L}(\mathcal{P}) = \mathbf{L}(G_{\mathcal{P}})$.

PDA $\rightsquigarrow$ Grammatik

Satz: Für jeden PDA $\mathcal{P}$ gibt es eine kontextfreie Grammatik $G_{\mathcal{P}}$, so dass $\mathbf{L}(\mathcal{P}) = \mathbf{L}(G_{\mathcal{P}})$.

Beweisidee: Für jedes Paar von Zuständen q und r betrachten wir die Sprache $\mathbf{L_{q,r}}$, die durch $\mathcal{P}$ akzeptiert wird, wenn man mit leerem Keller in Zustand q beginnt und in Zustand r mit leerem Keller endet:

$$w \in \mathbf{L_{q,r}} \quad \text{genau dann, wenn} \quad \langle q, w, \epsilon \rangle \vdash^* \langle r, \epsilon, \epsilon \rangle$$

- $\mathbf{L_{q,r}}$ wird in der Grammatik durch eine Variable $V_{q,r}$ dargestellt.
- Wir modifizieren $\mathcal{P}$, so dass
  - $\mathcal{P}$ genau einen Startzustand q_0 und genau einen Endzustand q_f hat;
  - der Keller vor Erreichen von q_f geleert werden muss.

  $\rightsquigarrow$ Die gesuchte Sprache $\mathbf{L}(\mathcal{P})$ ist genau $\mathbf{L}_{q_0, q_f}$.

Quiz: Zustandspaare in Kellerautomaten

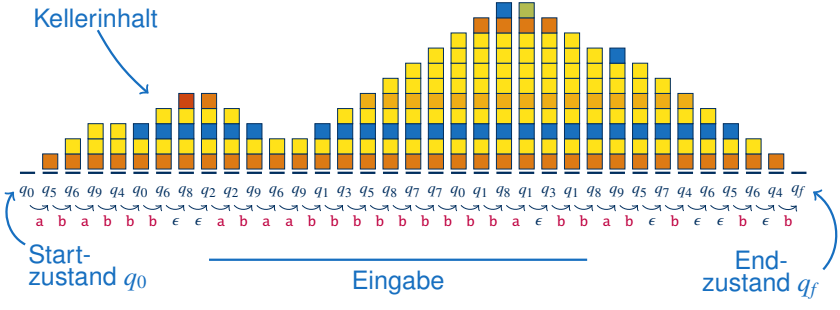
Für jedes Paar von Zuständen q und r betrachten wir die Sprache $\mathbf{L_{q,r}}$, die durch $\mathcal{M}$ akzeptiert wird, wenn man mit leerem Keller in Zustand q beginnt und in Zustand r mit leerem Keller endet: $w \in \mathbf{L_{q,r}}$ genau dann, wenn $\langle q, w, \epsilon \rangle \vdash^* \langle r, \epsilon, \epsilon \rangle$.

Quiz: Gegeben sei Kellerautomat $\mathcal{M} = \langle Q, \{\mathtt{0}, \mathtt{1}\}, \{\mathsf{S}, \mathsf{X}\}, \delta, \{q_0\}, \{q_3\} \rangle$ mit δ wie folgt:

...

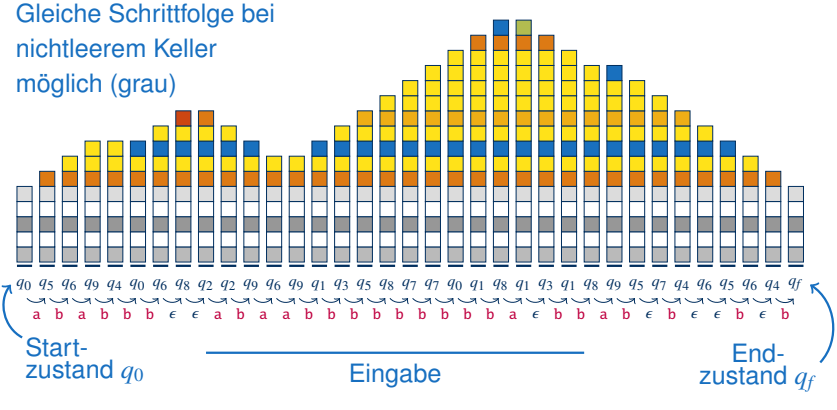
Die Sprache $\mathbf{L_{q,r}}$

Schematische Darstellung der Konfigurationsfolge bei der Akzeptanz eines Wortes aus $\mathbf{L}_{q_0, q_f}$:



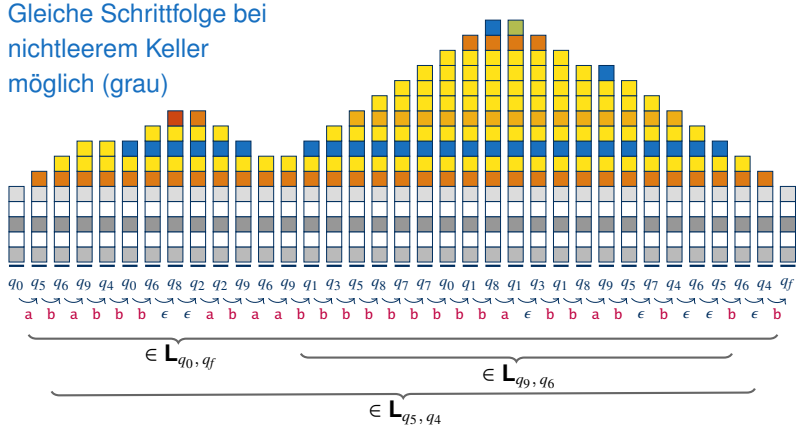
Die Sprache $\mathbf{L_{q,r}}$ (2)

Schematische Darstellung der Konfigurationsfolge bei der Akzeptanz eines Wortes aus $\mathbf{L}_{q_0, q_f}$:



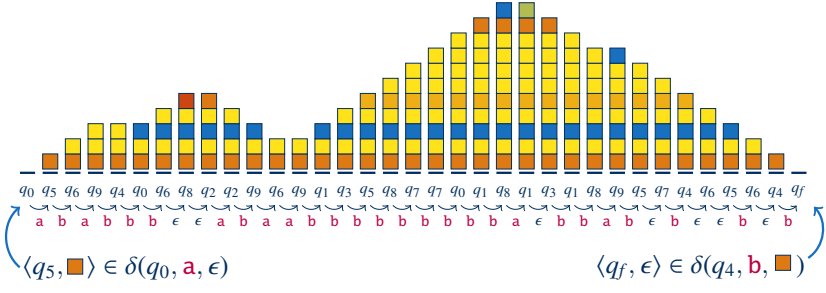
Die Sprache $\mathbf{L_{q,r}}$ (2)

Schematische Darstellung der Konfigurationsfolge bei der Akzeptanz eines Wortes aus $\mathbf{L}_{q_0, q_f}$:



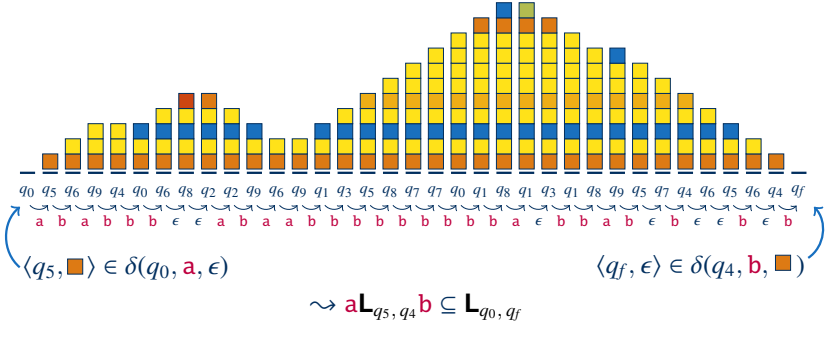
Die Sprache $\mathbf{L_{q,r}}$ (3)

Schematische Darstellung der Konfigurationsfolge bei der Akzeptanz eines Wortes aus $\mathbf{L}_{q_0, q_f}$:



Die Sprache $\mathbf{L_{q,r}}$ (3)

Schematische Darstellung der Konfigurationsfolge bei der Akzeptanz eines Wortes aus $\mathbf{L}_{q_0, q_f}$:



Eine Grammatik für $\mathbf{L_{q,r}}$

Wir wollen für jedes $\mathbf{L_{q,r}}$ eine Variable $V_{q,r}$ einführen, welche diese Sprache definiert

Aus dem gerade Gesehenen folgt:

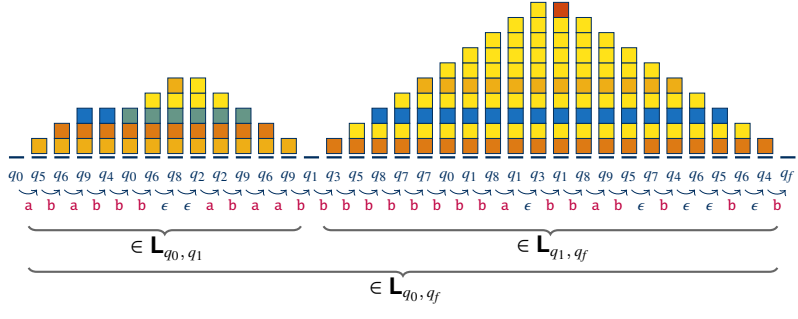
- Wenn es ein Kellersymbol A gibt,
- für das es ein Symbol a und einen Zustand s_1 gibt, so dass $\langle s_1, A\rangle \in \delta(q, a, \epsilon)$, und
- für das es ein Symbol b und einen Zustand s_2 gibt, so dass $\langle r, \epsilon\rangle \in \delta(s_2, b, A)$,
- dann sollte die Grammatik die Regel

$$V_{q,r} \to a V_{s_1,s_2} b$$

enthalten.

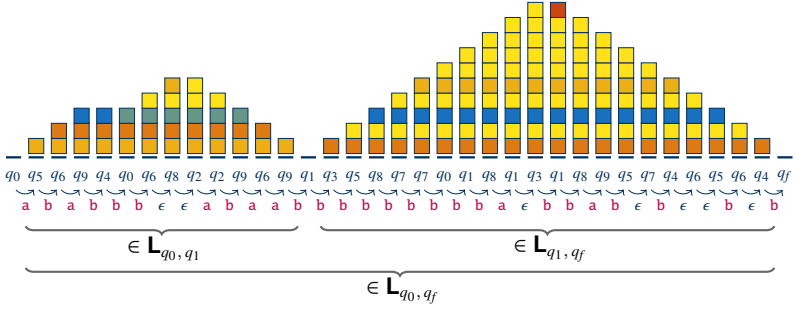
Die Sprache $\mathbf{L_{q,r}}$ (4)

Es gibt einen zweiten relevanten Fall zur Ableitung von $\mathbf{L}_{q_0, q_f}$:



Die Sprache $\mathbf{L_{q,r}}$ (4)

Es gibt einen zweiten relevanten Fall zur Ableitung von $\mathbf{L}_{q_0, q_f}$:



$$\rightsquigarrow \mathbf{L}_{q_0, q_1} \mathbf{L}_{q_1, q_f} \subseteq \mathbf{L}_{q_0, q_f}$$

Eine Grammatik für $\mathbf{L_{q,r}}$ (2)

Wir wollen für jedes $\mathbf{L_{q,r}}$ eine Variable $V_{q,r}$ einführen, welche diese Sprache definiert.

Aus dem gerade Gesehenen folgt:

- Für jeden Zustand s enthält die Grammatik die Regel

$$V_{q,r} \to V_{q,s} V_{s,r}$$

Außerdem benötigen wir noch die folgenden Regeln, damit die Ableitung irgendwann abgeschlossen werden kann:

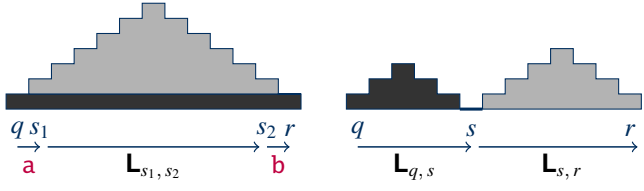
- Für jeden Zustand s enthält die Grammatik die Regel

$$V_{s,s} \to \epsilon$$

Zwischenbilanz

Bisher haben wir zwei Arten von Konfigurationsfolgen bzw. Grammatikregeln eingeführt:

$V_{q,r} \to a V_{s_1,s_2} b$ $\qquad$ $V_{q,r} \to V_{q,s} V_{s,r}$



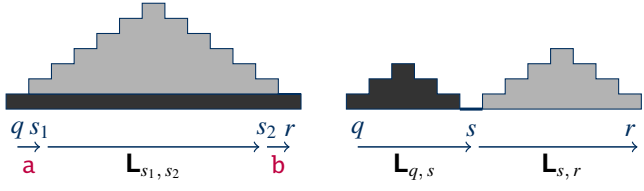
Man könnte auf diese beiden Arten einen beliebigen Lauf in immer kleinere Schritte zerlegen und somit das erkannte Wort generieren. Den Abschluss bilden die Regeln $V_{s,s} \to \epsilon$.

Funktioniert das im Prinzip für alle Läufe von $\mathcal{P}$?

Zwischenbilanz

Bisher haben wir zwei Arten von Konfigurationsfolgen bzw. Grammatikregeln eingeführt:

$V_{q,r} \to aV_{s_1,s_2}b$ $\qquad$ $V_{q,r} \to V_{q,s}V_{s,r}$



Man könnte auf diese beiden Arten einen beliebigen Lauf in immer kleinere Schritte zerlegen und somit das erkannte Wort generieren. Den Abschluss bilden die Regeln $V_{s,s} \to \epsilon$.

Funktioniert das im Prinzip für alle Läufe von $\mathcal{P}$?

Nein: Damit kann man Läufe generieren, bei denen sich die Höhe des Kellers in jedem Schritt ändert. Bleibt der Keller über mehrere Schritte gleich hoch, dann hilft keine der Zerlegungen weiter.

Abschluss des Beweises

Problem: Unsere Zerlegung funktioniert nicht, wenn der Keller über mehrere Schritte die gleiche Höhe behält.

Lösung: Wir können PDAs so abwandeln, dass sie in jedem Schritt entweder $\texttt{pop}$ oder $\texttt{push}$ (mindestens eines davon, aber nie beides) ausführen.

Skizze:

- Übergänge mit $\texttt{pop}$ und $\texttt{push}$ zerlegt man mithilfe eines Zwischenzustands in einen $\texttt{pop}$ und einen $\texttt{push}$;
- Übergänge ohne $\texttt{pop}$ oder $\texttt{push}$ stellt man dar, indem man zunächst ein Hilfszeichen $\texttt{push}t$ und dieses gleich darauf wieder $\texttt{pop}t$.

Mit diesen Änderungen gilt tatsächlich:

Fakt: $V_{q,r}$ erzeugt ein Wort w genau dann, wenn $\mathcal{P}$ von q mit leerem Keller zu r mit leerem Keller gelangen kann.

(ausführlicher Beweis siehe: Sipser, „Introduction to the Theory of Computation“ (Int. Ed.), Abschnitt 2.2.)

Der Beweis ist komplett

Zusammenfassung: Der PDA $\mathcal{P}$ wird so abgewandelt, dass:

- $\mathcal{P}$ genau einen Startzustand q_0 und genau einen Endzustand q_f hat;
- der Keller vor Erreichen von q_f geleert werden muss;
- $\mathcal{P}$ in jedem Schritt pop oder push ausführt, aber nie beides.

Die Grammatik $G(\mathcal{P})$ besteht aus den folgenden Regeln:

(1) Für alle Zustände $q, r, s_1, s_2 \in Q$, Terminale $\mathtt{a}, \mathtt{b} \in \Sigma_\epsilon$ und Kellersymbole $\mathtt{A} \in \Gamma$, für die es Übergänge $\langle s_1, \mathtt{A}\rangle \in \delta(q, \mathtt{a}, \epsilon)$ und $\langle r, \epsilon\rangle \in \delta(s_2, \mathtt{b}, \mathtt{A})$ gibt, die Regel $\mathtt{V}_{\mathtt{q},\mathtt{r}} \to \mathtt{a}\mathtt{V}_{\mathtt{s}_1,\mathtt{s}_2}\mathtt{b}$.

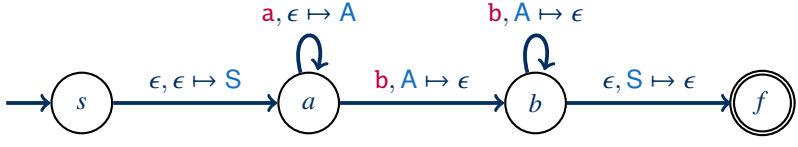
(2) Für alle Zustände $q, r, s \in Q$, die Regel $\mathtt{V}_{\mathtt{q},\mathtt{r}} \to \mathtt{V}_{\mathtt{q},\mathtt{s}}\mathtt{V}_{\mathtt{s},\mathtt{r}}$.

(3) Für alle Zustände $s \in Q$, die Regel $\mathtt{V}_{\mathtt{s},\mathtt{s}} \to \epsilon$.

Dann ist $\mathbf{L}(\mathcal{P}) = \mathbf{L}_{q_0, q_f}$. Die Grammatik $G(\mathcal{P})$ erzeugt diese Sprache, wenn wir $\mathtt{V}_{\mathtt{q}_0,\mathtt{q}_\mathtt{f}}$ als Startsymbol wählen. □

Beispiel

PDA für $\left\{ a^i b^i \;\middle|\; i \geq 1 \right\}$:



Dieser PDA erfüllt die Anforderungen des Beweises.

Als Grammatik ergibt sich:

$$V_{s,f} \to \epsilon V_{a,b} \epsilon$$

$$V_{a,b} \to a V_{a,b} b \mid a V_{a,a} b$$

$$V_{a,a} \to \epsilon \qquad V_{b,b} \to \epsilon \qquad V_{s,s} \to \epsilon \qquad V_{f,f} \to \epsilon$$

$$V_{a,a} \to V_{a,a} V_{a,a} \mid V_{a,b} V_{b,a} \mid V_{a,s} V_{s,a} \mid V_{a,f} V_{f,a} \qquad V_{a,b} \to \ldots \qquad \ldots$$

(Nicht alle dieser Regeln werden wirklich benötigt.)

Deterministische Kellerautomaten

Deterministische Kellerautomaten?

Es gibt verschiedene Quellen für Nichtdeterminismus bei PDAs:

Deterministische Kellerautomaten?

Es gibt verschiedene Quellen für Nichtdeterminismus bei PDAs:

- Die Übergangsfunktion liefert eine Menge möglicher Übergänge.

Deterministische Kellerautomaten?

Es gibt verschiedene Quellen für Nichtdeterminismus bei PDAs:

- Die Übergangsfunktion liefert eine Menge möglicher Übergänge.
- Es gibt mehrere mögliche Startzustände.

Deterministische Kellerautomaten?

Es gibt verschiedene Quellen für Nichtdeterminismus bei PDAs:

- Die Übergangsfunktion liefert eine Menge möglicher Übergänge.
- Es gibt mehrere mögliche Startzustände.
- Es gibt ϵ-Übergänge im Bezug auf Eingabe und Keller.

> **Beispiel:** Angenommen, es gilt $\delta(q, \epsilon, \mathsf{A}) = \{\langle p, \mathsf{B}\rangle\}$ und $\delta(q, \mathsf{a}, \epsilon) = \{\langle r, \mathsf{C}\rangle\}$. Dann gibt es mehrere mögliche Übergänge, wenn in Zustand q das Symbol a gelesen wird und A auf dem Keller liegt. Und das, obwohl die Übergangsfunktion nur einelementige Mengen liefert.

Deterministische Kellerautomaten müssen eingeschränkt werden, ohne ϵ-Übergänge ganz zu verbieten. (Völlig ohne ϵ-Übergänge sind PDAs zu schwach.)

Deterministische Kellerautomaten

Ein deterministischer Kellerautomat (international: „DPDA“) $\mathcal{P}$ ist ein Tupel $\mathcal{P} = \langle Q, \Sigma, \Gamma, \delta, q_0, F \rangle$ mit den folgenden Bestandteilen:

- Q: endliche Menge von Zuständen
- Σ: Eingabealphabet
- Γ: Kelleralphabet
- δ: Übergangsfunktion, eine partielle Funktion

$$Q \times \Sigma_\epsilon \times \Gamma_\epsilon \to Q \times \Gamma_\epsilon,$$

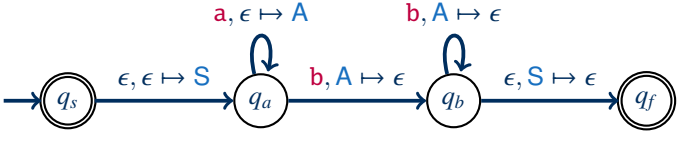
  so dass für alle $q \in Q$, $\mathsf{a} \in \Sigma$ und $\mathsf{A} \in \Gamma$
  jeweils nur eines der folgenden definiert ist:

$$\delta(q, \mathsf{a}, \mathsf{A}) \qquad \delta(q, \mathsf{a}, \epsilon) \qquad \delta(q, \epsilon, \mathsf{A}) \qquad \delta(q, \epsilon, \epsilon)$$

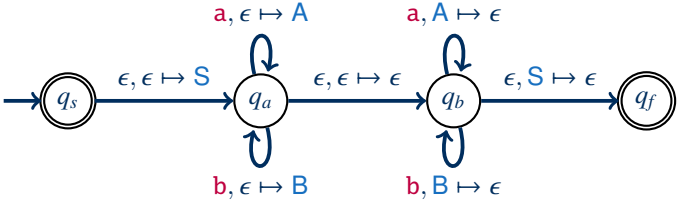
- q_0: ein Startzustand $q_0 \in Q$
- F: Menge von Endzuständen $F \subseteq Q$

Beispiele

Dieser Automat ist ein DPDA:



Dieser Automat ist kein DPDA:



Quiz: Deterministische PDAs

Quiz: Welche der folgenden grafisch gegebenen PDAs sind DPDAs? ...

Deterministische kontextfreie Sprachen

DPDAs führen zu einer eigenen Sprachklasse:

> Eine Sprache ist genau dann deterministisch kontextfrei, wenn sie durch einen deterministischen Kellerautomaten akzeptiert wird.

Offensichtlich ist jede deterministisch kontextfreie Sprache auch kontextfrei (DPDAs können als PDAs gesehen werden).

Deterministische kontextfreie Sprachen

DPDAs führen zu einer eigenen Sprachklasse:

Eine Sprache ist genau dann deterministisch kontextfrei, wenn sie durch einen deterministischen Kellerautomaten akzeptiert wird.

Offensichtlich ist jede deterministisch kontextfreie Sprache auch kontextfrei (DPDAs können als PDAs gesehen werden).

Im Gegensatz zur Situation bei DFAs und NFAs gilt die Umkehrung dieser Aussage nicht:

Satz: Die deterministisch kontextfreien Sprachen bilden eine echte Untermenge der kontextfreien Sprachen.

Es gibt also kontextfreie Sprachen, die nicht deterministisch kontextfrei sind.
(Diese Sprachen sind gewissermaßen „inhärent“ nichtdeterministisch.)

Abschluss unter Komplement

Der Unterschied zu allgemeinen kontextfreien Sprachen zeigt sich gut am folgenden Ergebnis:

Satz: Die Klasse der deterministisch kontextfreien Sprachen ist unter Komplement abgeschlossen.

Beweisidee: Wie bei DFAs kann man auch bei DPDAs die akzeptierenden und nichtakzeptierenden Zustände vertauschen.
Mehrere technische Komplikationen machen den Beweis etwas aufwändiger:

- Ein DPDA kann ein Wort auf zwei Arten nicht akzeptieren: (1) nach dem Lesen der Eingabe ist der Automat nicht in einem Endzustand; (2) die Eingabe wird nie ganz gelesen, z.B. weil der Automat in eine Endlosschleife von ϵ-Übergängen gerät. Man muss Grund (2) ausschalten.
- Ein Lauf könnte mit einer Folge von ϵ-Übergängen enden, die akzeptierende und nichtakzeptierende Zustände enthält. In diesem Fall führt Austausch der Endzustände nicht zum gewünschten Ergebnis. Man muss sicherstellen, dass grundsätzlich (vor und nach der Komplementierung) nur der erste Zustand in einer Folge von ϵ-Übergängen akzeptieren darf.

(vollständiger Beweis siehe Sipser, Abschnitt 2.4) □

Nichtdeterministische kontextfreie Sprachen

Eine kontextfreie Sprache, deren Komplement nicht kontextfrei ist, kann demnach nicht deterministisch kontextfrei sein.

Beispiel: In Vorlesung 14 haben wir die Sprachen $\mathbf{L_1} = \left\{ \mathtt{a}^i\mathtt{b}^i\mathtt{c}^k \;\middle|\; i \geq 0, k \geq 0 \right\}$ und $\mathbf{L_2} = \left\{ \mathtt{a}^i\mathtt{b}^k\mathtt{c}^k \;\middle|\; i \geq 0, k \geq 0 \right\}$ betrachtet und erkannt, dass die Sprache

$$\overline{\mathbf{L_1}} \cup \overline{\mathbf{L_2}} = \left\{ \mathtt{a}^i\mathtt{b}^j\mathtt{c}^k \;\middle|\; i \neq j \text{ oder } j \neq k \right\} \cup \left(\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}^* \circ \{\mathtt{ba}, \mathtt{ca}, \mathtt{cb}\} \circ \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}^* \right)$$

kontextfrei ist, aber ihr Komplement $\mathbf{L_1} \cap \mathbf{L_2}$ nicht.
Diese Sprache ist demnach nicht deterministisch kontextfrei.

Das beweist im wesentlichen die Behauptung, dass die deterministisch kontextfreien Sprachen eine echte Untermenge der kontextfreien Sprachen sind.

Nichtdeterministische kontextfreie Sprachen

Eine kontextfreie Sprache, deren Komplement nicht kontextfrei ist, kann demnach nicht deterministisch kontextfrei sein.

Beispiel: In Vorlesung 14 haben wir die Sprachen $\mathbf{L_1} = \left\{ \mathtt{a}^i\mathtt{b}^i\mathtt{c}^k \;\middle|\; i \geq 0, k \geq 0 \right\}$ und $\mathbf{L_2} = \left\{ \mathtt{a}^i\mathtt{b}^k\mathtt{c}^k \;\middle|\; i \geq 0, k \geq 0 \right\}$ betrachtet und erkannt, dass die Sprache

$$\overline{\mathbf{L_1}} \cup \overline{\mathbf{L_2}} = \left\{ \mathtt{a}^i\mathtt{b}^j\mathtt{c}^k \;\middle|\; i \neq j \text{ oder } j \neq k \right\} \cup \left(\{\mathtt{a},\mathtt{b},\mathtt{c}\}^* \circ \{\mathtt{ba},\mathtt{ca},\mathtt{cb}\} \circ \{\mathtt{a},\mathtt{b},\mathtt{c}\}^* \right)$$

kontextfrei ist, aber ihr Komplement $\mathbf{L_1} \cap \mathbf{L_2}$ nicht.
Diese Sprache ist demnach nicht deterministisch kontextfrei.

Das beweist im wesentlichen die Behauptung, dass die deterministisch kontextfreien Sprachen eine echte Untermenge der kontextfreien Sprachen sind.

Es gibt noch viele andere nichtdeterministisch kontextfreie Sprachen, z.B.:

Beispiel: Die Sprache $\mathbf{L_{pal}} = \{ww^r \mid w \in \{\mathtt{a},\mathtt{b}\}^*\}$ der Palindrome gerader Länge (ohne Markierung in der Mitte) ist kontextfrei, aber nicht deterministisch kontextfrei.

(Ohne Beweis; siehe z.B. Skript von Franz Baader)

Zusammenfassung und Ausblick

PDAs erkennen genau die kontextfreien Sprachen:

- Typ-2-Grammatik $\leadsto$ PDA durch Simulation von Linksableitungen
- PDA $\leadsto$ Typ-2-Grammatik durch rekursive Erzeugung der Sprachen $\mathbf{L_{q,r}}$

Deterministische Kellerautomaten (DPDAs) erkennen nur eine echte Untermenge der kontextfreien Sprachen.

Offene Fragen:

- Wozu sind deterministisch kontextfreie Sprachen gut?
- Welche Probleme auf CFGs kann man lösen?
- Was gibt es zu Typ 1 und Typ 0 zu sagen?