Formale Grundlagen der Informatik 1
Kapitel 4
Über reguläre Sprachen hinaus
Kellerautomaten und Pumping Lemma

Frank Heitmann
heitmann@informatik.uni-hamburg.de

20. April 2015

Grenzen regulärer Sprachen?

Wir haben mittlerweile einiges kennengelernt, um reguläre Sprachen zu erzeugen:

- direkte Konstruktion von (verschiedenen) Automaten
- direkte Konstruktion von rationalen Ausdrücken
- Benutzung von Abschlusseigenschaften

Als Anwendungsfälle hatten wir

- Worterkennung in Texten
- speziell im Compilerbau

Wiederholung

Wir wiederholen ...

- Mengen, Mengenoperationen etc. als Grundlage
- Alphabet, Wörter, Konkatenation, ...
- Darauf aufbauen: Sprachen

Damit dann:

- der **deterministische, endliche Automat** (DFA)

Wiederholung

Wir wiederholen ...

- Mengen, Mengenoperationen etc. als Grundlage
- Alphabet, Wörter, Konkatenation, ...
- Darauf aufbauen: Sprachen

Damit dann:

- der **deterministische, endliche Automat** (DFA)

Wiederholung

Wir wiederholen ...

- Mengen, Mengenoperationen etc. als Grundlage
- Alphabet, Wörter, Konkatenation, ...
- Darauf aufbauen: Sprachen

Damit dann:

- der **deterministische, endliche Automat** (DFA)

Wiederholung

Wir wiederholen ...

- Mengen, Mengenoperationen etc. als Grundlage
- Alphabet, Wörter, Konkatenation, ...
- Darauf aufbauen: Sprachen

Damit dann:

- der **deterministische, endliche Automat** (DFA)

Wiederholung

Wir wiederholen ...

- Mengen, Mengenoperationen etc. als Grundlage
- Alphabet, Wörter, Konkatenation, ...
- Darauf aufbauen: Sprachen

Damit dann:

- der **deterministische, endliche Automat** (DFA)

Der deterministische, endliche Automat

Definition (DFA)

Ein **deterministischer, endlicher Automat** (DFA) ist ein 5-Tupel

$$A = (Z, \Sigma, \delta, z_0, Z_{end})$$

mit:

- Der endlichen Menge von *Zuständen* Z.
- Dem endlichen Alphabet Σ von *Eingabesymbolen*.
- Der *Überführungsfunktion* $\delta : Z \times \Sigma \to Z$.
- Dem *Startzustand* $z_0 \in Z$.
- Der Menge der *Endzustände* $Z_{end} \subseteq Z$.

Arbeitsweise des DFA (informal)

Erhält ein DFA ein Eingabewort $w \in \Sigma^*$, so

- beginnt er im Startzustand z_0.
- Beginnt w mit dem Symbol $x \in \Sigma$, so
- wird der Nachfolgezustand nun durch $\delta(z_0, x)$ bestimmt.
- Dies wird dann in dem nun aktuellen Zustand und mit dem Restwort fortgeführt.
- Das Wort w wird akzeptiert, wenn
  - w bis zum Ende gelesen werden kann **und**
  - der Automat dann in einem Endzustand ist.

Arbeitsweise des DFA (informal)

Erhält ein DFA ein Eingabewort $w \in \Sigma^*$, so

- beginnt er im Startzustand z_0.
- Beginnt w mit dem Symbol $x \in \Sigma$, so
- wird der Nachfolgezustand nun durch $\delta(z_0, x)$ bestimmt.
- Dies wird dann in dem nun aktuellen Zustand und mit dem Restwort fortgeführt.
- Das Wort w wird akzeptiert, wenn
  - w bis zum Ende gelesen werden kann **und**
  - der Automat dann in einem Endzustand ist.

Arbeitsweise des DFA (informal)

Erhält ein DFA ein Eingabewort $w \in \Sigma^*$, so

- beginnt er im Startzustand z_0.
- Beginnt w mit dem Symbol $x \in \Sigma$, so
- wird der Nachfolgezustand nun durch $\delta(z_0, x)$ bestimmt.
- Dies wird dann in dem nun aktuellen Zustand und mit dem Restwort fortgeführt.
- Das Wort w wird akzeptiert, wenn
  - w bis zum Ende gelesen werden kann **und**
  - der Automat dann in einem Endzustand ist.

Arbeitsweise des DFA (informal)

Erhält ein DFA ein Eingabewort $w \in \Sigma^*$, so

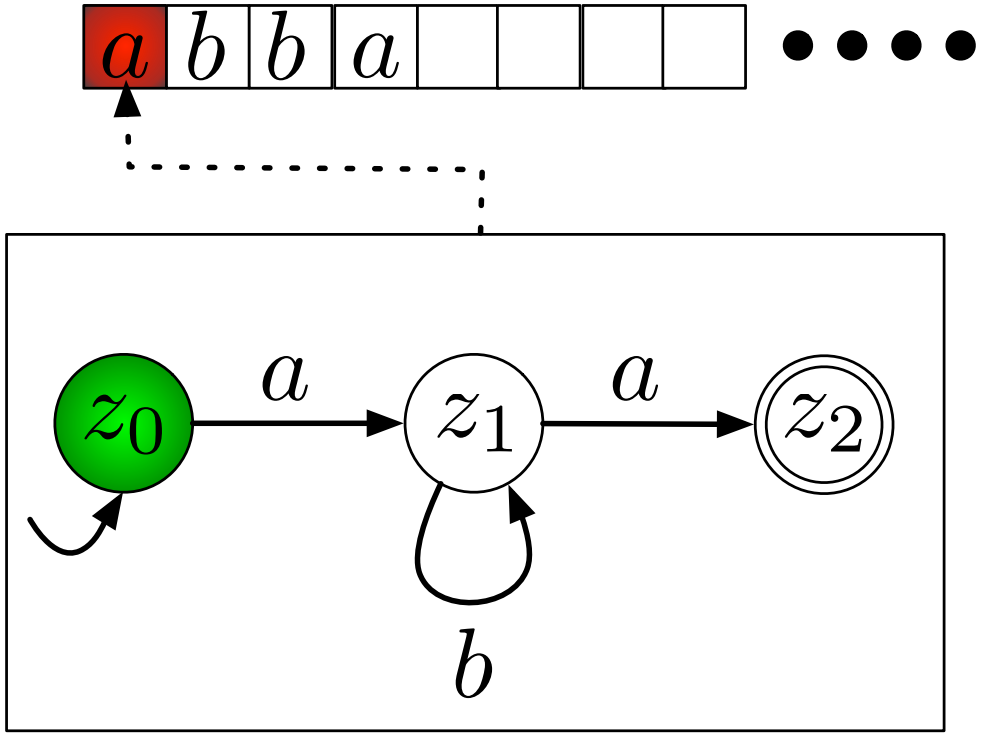
- beginnt er im Startzustand z_0.
- Beginnt w mit dem Symbol $x \in \Sigma$, so
- wird der Nachfolgezustand nun durch $\delta(z_0, x)$ bestimmt.
- Dies wird dann in dem nun aktuellen Zustand und mit dem Restwort fortgeführt.
- Das Wort w wird akzeptiert, wenn
  - w bis zum Ende gelesen werden kann **und**
  - der Automat dann in einem Endzustand ist.

Arbeitsweise des DFA (informal)

Erhält ein DFA ein Eingabewort $w \in \Sigma^*$, so

- beginnt er im Startzustand z_0.
- Beginnt w mit dem Symbol $x \in \Sigma$, so
- wird der Nachfolgezustand nun durch $\delta(z_0, x)$ bestimmt.
- Dies wird dann in dem nun aktuellen Zustand und mit dem Restwort fortgeführt.
- Das Wort w wird akzeptiert, wenn
  - w bis zum Ende gelesen werden kann **und**
  - der Automat dann in einem Endzustand ist.

Ein Beispiel (DFA)

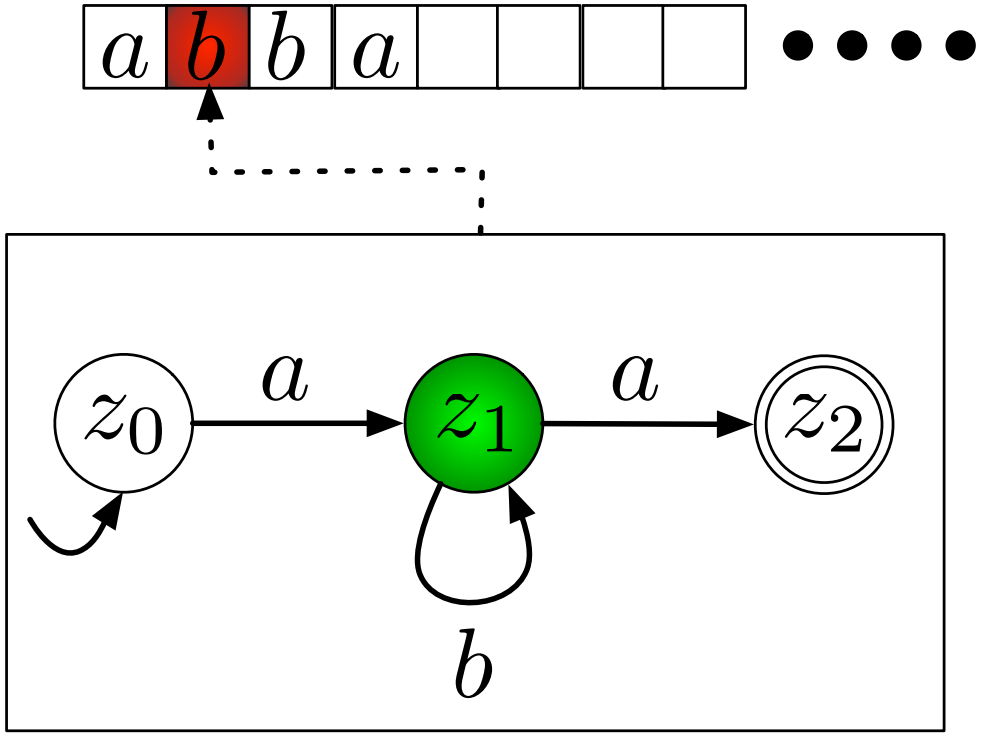


$(z_0, abba)$

oder

$\hat{\delta}(z_0, abba)$

Ein Beispiel (DFA)

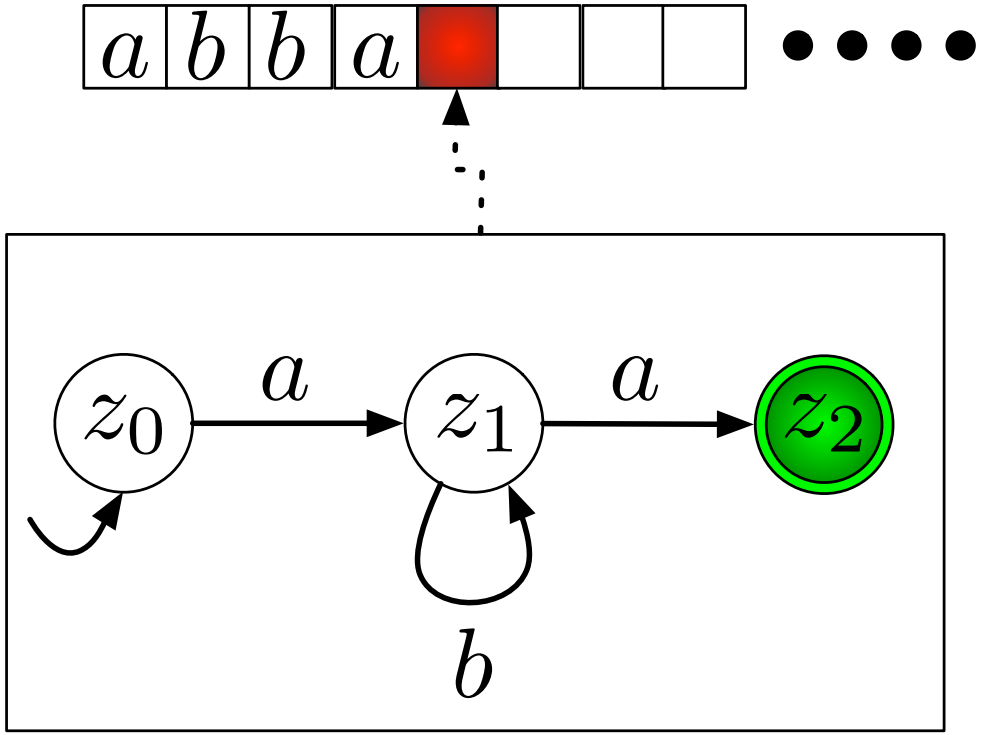


$$(z_0, abba) \vdash (z_1, bba)$$

oder

$$\hat{\delta}(z_0, abba) = \hat{\delta}(z_1, bba)$$

Ein Beispiel (DFA)



$$(z_0, abba) \vdash (z_1, bba) \vdash (z_1, ba) \vdash (z_1, a) \vdash (z_2, \lambda)$$

oder

$$\hat{\delta}(z_0, abba) = \hat{\delta}(z_1, bba) = \hat{\delta}(z_1, ba) = \hat{\delta}(z_1, a) = z_2$$

Akzeptierte Sprache (DFA)

Definition (Akzeptierte Sprache)

Die von einem DFA A **akzeptierte Sprache** ist die Menge

$$\begin{aligned} L(A) &:= \{w \in \Sigma^* \mid \hat{\delta}(z_0, w) \in Z_{end}\} \\ &= \{w \in \Sigma^* \mid (z_0, w) \vdash^* (z_e, \lambda), z_e \in Z_{end}\} \end{aligned}$$

Diese Menge wird auch als **reguläre Menge** bezeichnet. Die **Familie aller regulären Mengen** wird mit REG bezeichnet.

Wichtige Anmerkung

A akzeptiert ein Wort w genau dann, wenn w bis zum Ende gelesen werden kann **und** er dann in einem Endzustand ist.

Der nichtdeterministische, endliche Automat

Definition (NFA)

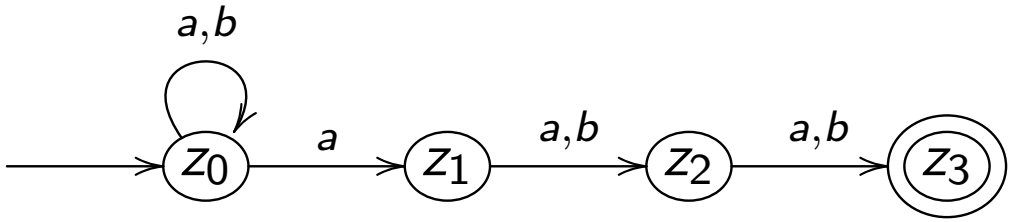
Ein **nichtdeterministischer, endlicher Automat** (NFA) ist ein 5-Tupel

$$A = (Z, \Sigma, \delta, Z_{start}, Z_{end})$$

mit:

- Der endlichen Menge von *Zuständen* Z.
- Dem endlichen Alphabet Σ von *Eingabesymbolen*.
- Der *Überführungsfunktion* $\delta : Z \times \Sigma \to 2^Z$.
- Der Menge der *Startzustände* $Z_{start} \subseteq Z$.
- Der Menge der *Endzustände* $Z_{end} \subseteq Z$.

Ein Beispiel (NFA)



Zur Arbeitsweise

Der NFA arbeitet in einer Rechnung so wie ein DFA, kann aber in jedem Schritt nichtdeterministisch in mehrere Zustände wechseln.

Akzeptierte Sprache (NFA)

Definition (Akzeptierte Sprache)

Die von einem NFA A **akzeptierte Sprache** ist die Menge

$$\begin{aligned} L(A) &:= \{w \in \Sigma^* \mid \hat{\delta}(Z_{start}, w) \in Z_{end}\} \\ &= \{w \in \Sigma^* \mid (z_0, w) \vdash^* (z_e, \lambda), z_0 \in Z_{start}, z_e \in Z_{end}\} \end{aligned}$$

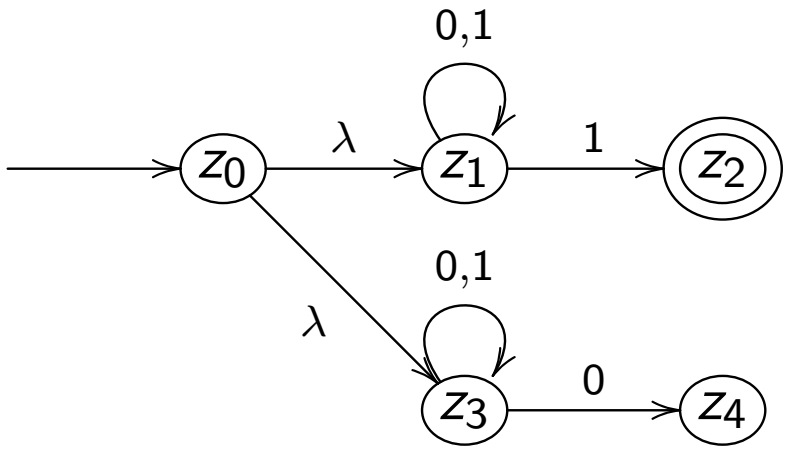
Wichtige Anmerkung

Der NFA akzeptiert, sobald es auf einem Wort **mindestens eine akzeptierende Rechnung** gibt.

NFAs mit Lambda-Kanten

Man kann einem NFA zusätzlich noch λ-Kanten (oder ϵ-Kanten) erlauben:

- nichts vom Eingabeband lesen
- es findet nur ein Zustandswechsel statt



Reguläre Ausdrücke

Definition (Reguläre Ausdrücke)

Sei Σ ein Alphabet. Die **regulären Ausdrücke über** Σ sind induktiv definiert durch:

1. $\emptyset$ ist ein regulärer Ausdruck, der die Menge $M_\emptyset = \emptyset$ beschreibt.
2. Für jedes $a \in \Sigma$ ist a ein regulärer Ausdruck, der die Menge $M_a = \{a\}$ beschreibt.
3. Sind X und Y reguläre Ausdrücke, die die Mengen M_X und M_Y beschreiben, dann beschreibt
   - $(X + Y)$ die Menge $M_X \cup M_Y$
   - $(X \cdot Y)$ die Menge $M_X \cdot M_Y$
   - X^* die Menge M_X^* und
   - X^+ die Menge M_X^+.
4. Nur die so erzeugten Ausdrücke sind reguläre Ausdrücke.

Eine Technik

Wichtiges Vorgehen

Ermittelt man für einen DFA A seine akzeptierte Sprache M bzw. konstruiert man zu einer Sprache M einen DFA A, so ist $L(A) = M$ zunächst eine Behauptung, die zu zeigen ist!

Wichtiges Vorgehen

Hierzu sind dann **zwei Richtungen** zu zeigen:

- $L(A) \subseteq M$.
- $M \subseteq L(A)$.

Wichtiges Vorgehen

Dies gilt ganz entsprechend für NFAs und reguläre Ausdrücke.

Reguläre Sprachen

Zusammenfassung

Zu jeder regulären Sprache L gibt es

- einen DFA A mit $L(A) = L$
- einen NFA B mit $L(B) = L$
- einen NFA mit λ-Kanten C mit $L(C) = L$
- einen regulären Ausdruck D, der L beschreibt ($M_D = L$).

DFAs, NFAs, NFAs mit λ-Kanten und reguläre Ausdrücke sind also äquivalent **und beschreiben die Familie der regulären Sprachen**.

Der Potenzautomat

Satz

Zu jeder von einem NFA A akzeptierte Menge L kann ein DFA B konstruiert werden mit $L(B) = L$.

Die Konstruktion

Sei $A = (Z, \Sigma, \delta, Z_{start}, Z_{end})$ der NFA. Wir definieren $B = (Z', \Sigma', \delta', z_0, Z'_{end})$ mit

1. $Z' := 2^Z$
2. $\Sigma' := \Sigma$
3. $\delta'(M, x) := \cup_{z \in M} \delta(z, x)$ oder alternativ
   $\delta'(M, x) := \cup_{z \in M} \{z' \in Z \mid (z, x, z') \in K\}$
4. $z_0 := Z_{start}$
5. $Z'_{end} := \{M \in 2^Z \mid M \cap Z_{end} \neq \emptyset\}$

Abschlusseigenschaften

Definition

Sei f_1 eine einstellige Operation auf Mengen und f_2 eine zweistellige Operationen. D.h. wenn M_1, M_2 zwei Mengen sind, dann sind auch $f_1(M_1)$ und $f_2(M_1, M_2)$ Mengen.

- Eine Sprachfamilie $\mathcal{C}$ ist **abgeschlossen** gegenüber der Operation f_1 bzw. f_2, wenn für jedes $R \in \mathcal{C}$ auch $f_1(R) \in \mathcal{C}$ gilt bzw. wenn für $R_1, R_2 \in \mathcal{C}$ auch $f_2(R_1, R_2) \in \mathcal{C}$ gilt.

Beispiel

Oft schreibt man die Operatoren in Infix-Notation (d.h. den Operator zwischen die Argumente). Beispiele in einem anderen Kontext sind z.B. die Addition bei den natürlichen Zahlen.

Abschlusseigenschaften - Zusammenfassung

Satz

Die regulären Sprachen sind abgeschlossen gegenüber

1. *Vereinigung* $\cup$, *Konkatenation* $\cdot$, *"hoch* $+$*"*, *"hoch* $*$*"*
2. *Komplementbildung*, *Durchschnitt* $\cap$
3. ...

Beweis.

1. Mit regulären Ausdrücken
2. Mit (vollständigen) DFAs
3. ...

$\square$

Zusammenfassung - Begriffe

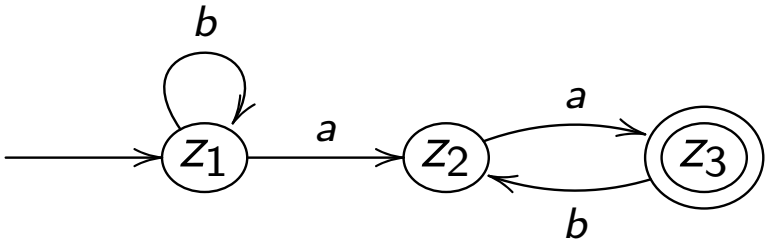
- DFA, NFA, (NFA mit λ-Kanten)
  - Zustände, Startzustand, Endzustände
  - Überführungsfunktion
  - erweitere Überführungsfunktion
  - vollständig, initial zusammenhängend
  - Konfiguration, Konfigurationsübergang
  - Rechnung, Erfolgsrechnung
  - akzeptierte Sprache
- reguläre Ausdrücke
- Potenzautomatenkonstruktion
- Konstruktionstechniken und $L(A) = M$
- Abschlusseigenschaften von *Reg*

Fragen...

Sei $L \subseteq \Sigma^*$ eine Sprache. Ist dann $\Sigma^* \setminus L$ auch eine Sprache? Falls ja, ist sie eine reguläre Sprache?

1. Nein und Nein
2. Ja und Nein
3. Ja und Manchmal
4. Ja und Ja

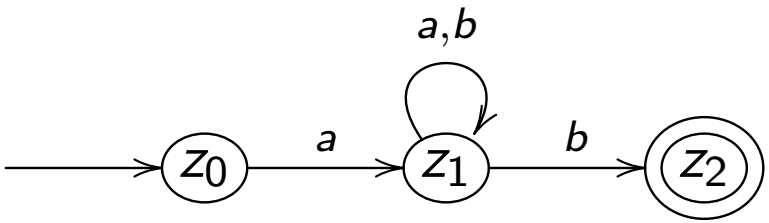
Fragen...



Welche Sprache akzeptiert obiger DFA?

1. b^*aa
2. $b^*aa(ba)^*$
3. $b^*(a+b)^*$
4. $b^+aa(ba)^+$

Fragen...



Was ist $\hat{\delta}(\{z_1, z_2\}, b)$ und $\hat{\delta}(\{z_0, z_1\}, ab)$?

1. $\{z_0, z_1\}$ und $\{z_1, z_2\}$
2. $\{z_1, z_2\}$ und $\{z_1\}$
3. $\{z_1\}$ und $\{z_2\}$
4. $\{z_1, z_2\}$ und $\{z_1, z_2\}$

Fragen...

Sie A ein NFA mit 6 Zuständen. Wie viele Zustände hat ein durch die Potenzautomatenkonstruktion gewonnener DFA maximal?

1. 6
2. 12
3. 36
4. 64

Fragen...

Sei A ein NFA mit 5 Zuständen. Wie viele Zustände hat ein äquivalenter DFA mindestens?

1. 1
2. 5
3. 25
4. 32

Fragen...

Welcher reguläre Ausdruck beschreibt die Menge $\overline{\{a\}^*}$?

1. $(a + b)^* \cdot b \cdot (a + b)^*$
2. b^*
3. $(a + b)^* \setminus a^*$
4. $(a + b)^* \cdot b^* \cdot (a + b)^*$

Zur Nachbereitung

Zur Nachbereitung

Richtige Antworten sind:

1. 3
2. 2
3. 4
4. 4
5. 1
6. 1

Grenzen regulärer Sprachen?

Und obwohl nun viele Sprachen regulär sind, gibt es (gerade auch in obigem Kontext) welche, bei denen es uns schwerfällt einen Automaten zu konstruieren.

Beispiel

Die Sprache aller korrekt geklammerten Ausdrücke ?

Z.B. wollen wir ()(()) akzeptieren, (() nicht.

Grenzen regulärer Sprachen?

Und obwohl nun viele Sprachen regulär sind, gibt es (gerade auch in obigem Kontext) welche, bei denen es uns schwerfällt einen Automaten zu konstruieren.

Beispiel

Die Sprache aller korrekt geklammerten Ausdrücke ?

Z.B. wollen wir ()(()) akzeptieren, (() nicht.

Grenzen regulärer Sprachen?

Mit unseren bisherigen Techniken scheitern wir hier...
... geht das überhaupt?!

Eine einfachere Sprache, die aber einen wichtigen Aspekt erfasst:

$$\{a^n b^n \mid n \in \mathbb{N}\}$$

Ist diese regulär ?

Wir scheitern zunächst auch hier ...

Grenzen regulärer Sprachen?

Mit unseren bisherigen Techniken scheitern wir hier...
... geht das überhaupt?!

Eine einfachere Sprache, die aber einen wichtigen Aspekt erfasst:

$$\{a^n b^n \mid n \in \mathbb{N}\}$$

Ist diese regulär ?

Wir scheitern zunächst auch hier ...

Grenzen regulärer Sprachen?

Mit unseren bisherigen Techniken scheitern wir hier...
... geht das überhaupt?!

Eine einfachere Sprache, die aber einen wichtigen Aspekt erfasst:

$$\{a^n b^n \mid n \in \mathbb{N}\}$$

Ist diese regulär ?

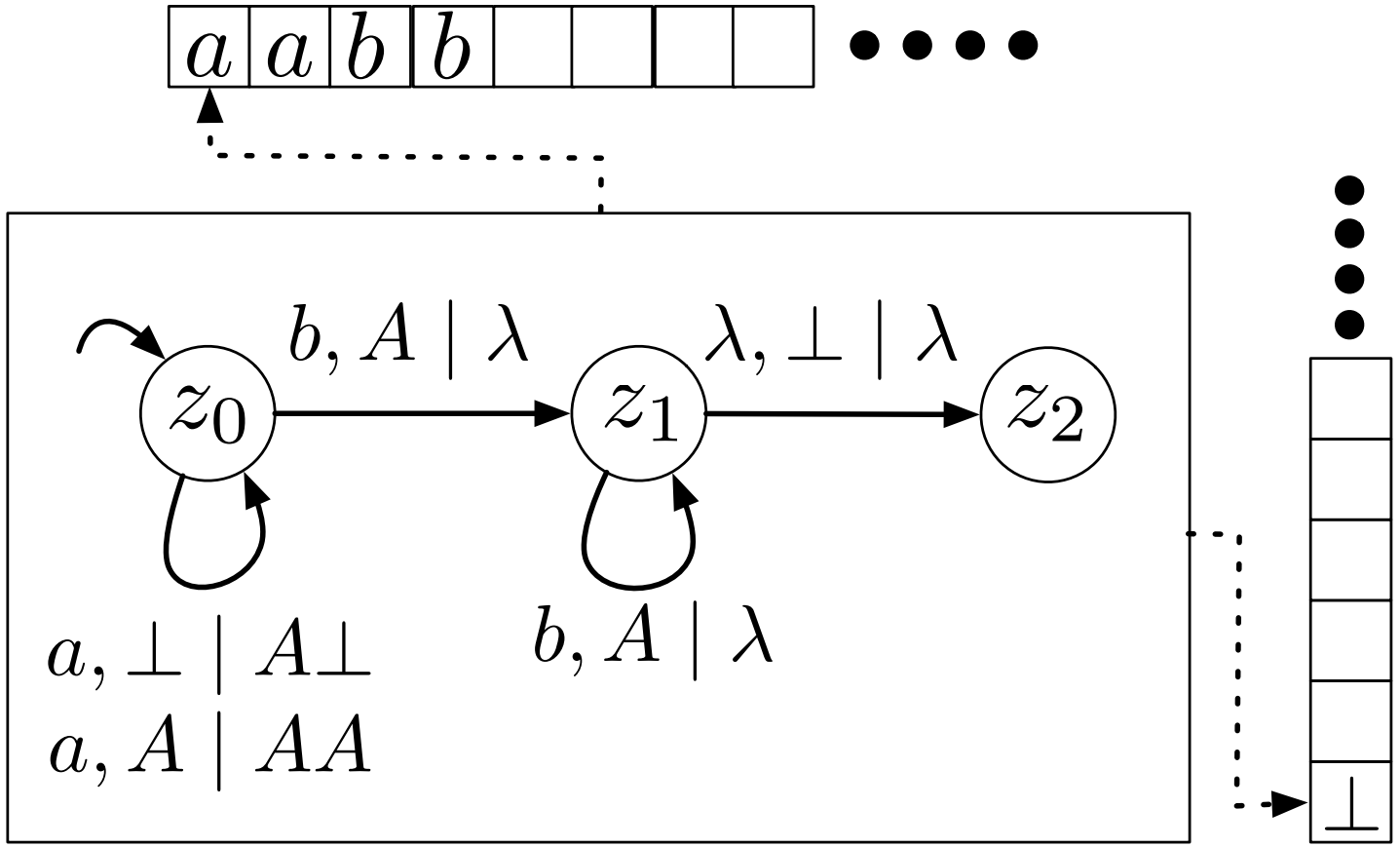
Wir scheitern zunächst auch hier ...

Kellerautomaten

Wir erweitern den DFA um einen Keller, um uns Dinge zu merken.

Den so entstehenden Automaten nennen wir **Kellerautomat**.

Der Kellerautomat



Der Kellerautomat - Formal

Definition (Kellerautomat (PDA = Push Down Automata))

Ein **nichtdeterministischer Kellerautomat** (kurz PDA) ist ein 7-Tupel $A = (Z, \Sigma, \Gamma, \delta, Z_{start}, Z_{end}, \bot)$ mit

- Der endlichen Menge von *Zuständen* Z.
- Dem endlichen Alphabet Σ von *Eingabesymbolen*.
- Dem endlichen Alphabet Γ von *Kellersymbolen*.
- Der *Überführungsfunktion* $\delta : Z \times (\Sigma \cup \{\lambda\}) \times \Gamma \to 2^{Z \times \Gamma^*}$.
- Der Menge von *Startzuständen* $Z_{start} \subseteq Z$.
- Der Menge der *Endzustände* $Z_{end} \subseteq Z$.
- Dem *Kellerbodensymbol* $\bot \in \Gamma$.

Anmerkung

Dieses Automatenmodell ist *nichtdeterministisch*

Der Kellerautomat - Formal

Definition (Kellerautomat (PDA = Push Down Automata))

Ein **nichtdeterministischer Kellerautomat** (kurz PDA) ist ein 7-Tupel $A = (Z, \Sigma, \Gamma, \delta, Z_{start}, Z_{end}, \bot)$ mit

- Der endlichen Menge von *Zuständen* Z.
- Dem endlichen Alphabet Σ von *Eingabesymbolen*.
- Dem endlichen Alphabet Γ von *Kellersymbolen*.
- Der *Überführungsfunktion* $\delta : Z \times (\Sigma \cup \{\lambda\}) \times \Gamma \to 2^{Z \times \Gamma^*}$.
- Der Menge von *Startzuständen* $Z_{start} \subseteq Z$.
- Der Menge der *Endzustände* $Z_{end} \subseteq Z$.
- Dem *Kellerbodensymbol* $\bot \in \Gamma$.

Anmerkung

Dieses Automatenmodell ist *nichtdeterministisch*

Der Kellerautomat - Formal

Definition (Kellerautomat (PDA = Push Down Automata))

Ein **nichtdeterministischer Kellerautomat** (kurz PDA) ist ein 7-Tupel $A = (Z, \Sigma, \Gamma, \delta, Z_{start}, Z_{end}, \bot)$ mit

- Der endlichen Menge von *Zuständen* Z.
- Dem endlichen Alphabet Σ von *Eingabesymbolen*.
- Dem endlichen Alphabet Γ von *Kellersymbolen*.
- Der *Überführungsfunktion* $\delta : Z \times (\Sigma \cup \{\lambda\}) \times \Gamma \to 2^{Z \times \Gamma^*}$.
- Der Menge von *Startzuständen* $Z_{start} \subseteq Z$.
- Der Menge der *Endzustände* $Z_{end} \subseteq Z$.
- Dem *Kellerbodensymbol* $\bot \in \Gamma$.

Anmerkung

Dieses Automatenmodell ist *nichtdeterministisch*

Der Kellerautomat - Formal

Definition (Kellerautomat (PDA = Push Down Automata))

Ein **nichtdeterministischer Kellerautomat** (kurz PDA) ist ein 7-Tupel $A = (Z, \Sigma, \Gamma, \delta, Z_{start}, Z_{end}, \bot)$ mit

- Der endlichen Menge von *Zuständen* Z.
- Dem endlichen Alphabet Σ von *Eingabesymbolen*.
- Dem endlichen Alphabet Γ von *Kellersymbolen*.
- Der *Überführungsfunktion* $\delta : Z \times (\Sigma \cup \{\lambda\}) \times \Gamma \to 2^{Z \times \Gamma^*}$.
- Der Menge von *Startzuständen* $Z_{start} \subseteq Z$.
- Der Menge der *Endzustände* $Z_{end} \subseteq Z$.
- Dem *Kellerbodensymbol* $\bot \in \Gamma$.

Anmerkung

Dieses Automatenmodell ist *nichtdeterministisch*

Der Kellerautomat - Formal

Definition (Kellerautomat (PDA = Push Down Automata))

Ein **nichtdeterministischer Kellerautomat** (kurz PDA) ist ein 7-Tupel $A = (Z, \Sigma, \Gamma, \delta, Z_{start}, Z_{end}, \bot)$ mit

- Der endlichen Menge von *Zuständen* Z.
- Dem endlichen Alphabet Σ von *Eingabesymbolen*.
- Dem endlichen Alphabet Γ von *Kellersymbolen*.
- Der *Überführungsfunktion* $\delta : Z \times (\Sigma \cup \{\lambda\}) \times \Gamma \to 2^{Z \times \Gamma^*}$.
- Der Menge von *Startzuständen* $Z_{start} \subseteq Z$.
- Der Menge der *Endzustände* $Z_{end} \subseteq Z$.
- Dem *Kellerbodensymbol* $\bot \in \Gamma$.

Anmerkung

Dieses Automatenmodell ist *nichtdeterministisch*

Der Kellerautomat - Formal

Definition (Kellerautomat (PDA = Push Down Automata))

Ein **nichtdeterministischer Kellerautomat** (kurz PDA) ist ein 7-Tupel $A = (Z, \Sigma, \Gamma, \delta, Z_{start}, Z_{end}, \bot)$ mit

- Der endlichen Menge von *Zuständen* Z.
- Dem endlichen Alphabet Σ von *Eingabesymbolen*.
- Dem endlichen Alphabet Γ von *Kellersymbolen*.
- Der *Überführungsfunktion* $\delta : Z \times (\Sigma \cup \{\lambda\}) \times \Gamma \to 2^{Z \times \Gamma^*}$.
- Der Menge von *Startzuständen* $Z_{start} \subseteq Z$.
- Der Menge der *Endzustände* $Z_{end} \subseteq Z$.
- Dem *Kellerbodensymbol* $\bot \in \Gamma$.

Anmerkung

Dieses Automatenmodell ist *nichtdeterministisch*

Der Kellerautomat - Formal

Definition (Kellerautomat (PDA = Push Down Automata))

Ein **nichtdeterministischer Kellerautomat** (kurz PDA) ist ein 7-Tupel $A = (Z, \Sigma, \Gamma, \delta, Z_{start}, Z_{end}, \bot)$ mit

- Der endlichen Menge von *Zuständen* Z.
- Dem endlichen Alphabet Σ von *Eingabesymbolen*.
- Dem endlichen Alphabet Γ von *Kellersymbolen*.
- Der *Überführungsfunktion* $\delta : Z \times (\Sigma \cup \{\lambda\}) \times \Gamma \to 2^{Z \times \Gamma^*}$.
- Der Menge von *Startzuständen* $Z_{start} \subseteq Z$.
- Der Menge der *Endzustände* $Z_{end} \subseteq Z$.
- Dem *Kellerbodensymbol* $\bot \in \Gamma$.

Anmerkung

Dieses Automatenmodell ist *nichtdeterministisch*

Der Kellerautomat - Formal

Definition (Kellerautomat (PDA = Push Down Automata))

Ein **nichtdeterministischer Kellerautomat** (kurz PDA) ist ein 7-Tupel $A = (Z, \Sigma, \Gamma, \delta, Z_{start}, Z_{end}, \bot)$ mit

- Der endlichen Menge von *Zuständen* Z.
- Dem endlichen Alphabet Σ von *Eingabesymbolen*.
- Dem endlichen Alphabet Γ von *Kellersymbolen*.
- Der *Überführungsfunktion* $\delta : Z \times (\Sigma \cup \{\lambda\}) \times \Gamma \to 2^{Z \times \Gamma^*}$.
- Der Menge von *Startzuständen* $Z_{start} \subseteq Z$.
- Der Menge der *Endzustände* $Z_{end} \subseteq Z$.
- Dem *Kellerbodensymbol* $\bot \in \Gamma$.

Anmerkung

Dieses Automatenmodell ist *nichtdeterministisch*

Der Kellerautomat - Formal

Wie beim NFA kann statt der Überführungsfunktion

$$\delta : Z \times (\Sigma \cup \{\lambda\}) \times \Gamma \to 2^{Z \times \Gamma^*}$$

auch mit einer Relation

$$K \subseteq Z \times (\Sigma \cup \{\lambda\}) \times \Gamma \times \Gamma^* \times Z$$

gearbeitet werden.

Konfiguration und Rechnung

Definition (Konfiguration und Rechnung)

Eine **Konfiguration** eines PDA A ist ein Element

$$(z, w, v) \in Z \times \Sigma^* \times \Gamma^*,$$

mit der Bedeutung,

- dass A im Zustand z ist,
- das w noch auf dem Eingabeband zu lesen ist und
- der aktuelle Kellerinhalt v ist.

Die **Überführungsrelation** $\vdash \subseteq (Z \times \Sigma^* \times \Gamma^*) \times (Z \times \Sigma^* \times \Gamma^*)$ ist definiert durch

$$(z, xu, yw) \vdash (z', u, y'w) \text{ gdw. } (z', y') \in \delta(z, x, y).$$

Eine **Rechnung** ist wieder eine Folge solcher Übergänge.

Konfiguration und Rechnung

Definition (Konfiguration und Rechnung)

Eine **Konfiguration** eines PDA A ist ein Element

$$(z, w, v) \in Z \times \Sigma^* \times \Gamma^*,$$

mit der Bedeutung,

- dass A im Zustand z ist,
- das w noch auf dem Eingabeband zu lesen ist und
- der aktuelle Kellerinhalt v ist.

Die **Überführungsrelation** $\vdash \subseteq (Z \times \Sigma^* \times \Gamma^*) \times (Z \times \Sigma^* \times \Gamma^*)$ ist definiert durch

$$(z, xu, yw) \vdash (z', u, y'w) \text{ gdw. } (z', y') \in \delta(z, x, y).$$

Eine **Rechnung** ist wieder eine Folge solcher Übergänge.

Konfiguration und Rechnung

Definition (Konfiguration und Rechnung)

Eine **Konfiguration** eines PDA A ist ein Element

$$(z, w, v) \in Z \times \Sigma^* \times \Gamma^*,$$

mit der Bedeutung,

- dass A im Zustand z ist,
- das w noch auf dem Eingabeband zu lesen ist und
- der aktuelle Kellerinhalt v ist.

Die **Überführungsrelation** $\vdash \subseteq (Z \times \Sigma^* \times \Gamma^*) \times (Z \times \Sigma^* \times \Gamma^*)$ ist definiert durch

$$(z, xu, yw) \vdash (z', u, y'w) \text{ gdw. } (z', y') \in \delta(z, x, y).$$

Eine **Rechnung** ist wieder eine Folge solcher Übergänge.

Konfiguration und Rechnung

Definition (Konfiguration und Rechnung)

Eine **Konfiguration** eines PDA A ist ein Element

$$(z, w, v) \in Z \times \Sigma^* \times \Gamma^*,$$

mit der Bedeutung,

- dass A im Zustand z ist,
- das w noch auf dem Eingabeband zu lesen ist und
- der aktuelle Kellerinhalt v ist.

Die **Überführungsrelation** $\vdash \subseteq (Z \times \Sigma^* \times \Gamma^*) \times (Z \times \Sigma^* \times \Gamma^*)$ ist definiert durch

$$(z, xu, yw) \vdash (z', u, y'w) \text{ gdw. } (z', y') \in \delta(z, x, y).$$

Eine **Rechnung** ist wieder eine Folge solcher Übergänge.

Akzeptierte Sprache

Definition (Akzeptierte Sprache)

Die von einem PDA *mit leerem Keller akzeptierte Sprache* ist die Menge

$$L_\lambda(A) := \{w \in \Sigma^* \mid (z_0, w, \bot) \vdash^* (z, \lambda, \lambda)\}$$

und die von einem PDA *mit Endzustand akzeptierte Sprache* ist die Menge

$$L_Z(A) := \{w \in \Sigma^* \mid (z_0, w, \bot) \vdash^* (z_e, \lambda, v), z_e \in Z_{end}, v \in \Gamma^*\}.$$

Anmerkung

Anmerkungen auf der nächsten Folie!

Akzeptierte Sprache

Definition (Akzeptierte Sprache)

Die von einem PDA *mit leerem Keller akzeptierte Sprache* ist die Menge

$$L_\lambda(A) := \{w \in \Sigma^* \mid (z_0, w, \bot) \vdash^* (z, \lambda, \lambda)\}$$

und die von einem PDA *mit Endzustand akzeptierte Sprache* ist die Menge

$$L_Z(A) := \{w \in \Sigma^* \mid (z_0, w, \bot) \vdash^* (z_e, \lambda, v), z_e \in Z_{end}, v \in \Gamma^*\}.$$

Anmerkung

Anmerkungen auf der nächsten Folie!

Akzeptierte Sprache

$$L_\lambda(A) := \{w \in \Sigma^* \mid (z_0, w, \bot) \vdash^* (z, \lambda, \lambda)\}$$
$$L_Z(A) := \{w \in \Sigma^* \mid (z_0, w, \bot) \vdash^* (z_e, \lambda, v), z_e \in Z_{end}, v \in \Gamma^*\}$$

Anmerkung

- Bei $L_\lambda(A)$ werden die Endzustände nicht benutzt und
- bei $L_Z(A)$ ist der Kellerinhalt v am Ende egal.
- Statt $L_\lambda(A)$ schreiben wir auch (insb. im Skript) $N(A)$ und statt $L_Z(A)$ auch $L(A)$.
- Da das Modell nichtdeterministisch ist, genügt es wie beim NFA, wenn es bei einem Eingabewort **eine Rechnung gibt, die den Keller leert bzw. den Automaten in einen Endzustand bringt**.

Akzeptierte Sprache

$$L_\lambda(A) := \{w \in \Sigma^* \mid (z_0, w, \bot) \vdash^* (z, \lambda, \lambda)\}$$
$$L_Z(A) := \{w \in \Sigma^* \mid (z_0, w, \bot) \vdash^* (z_e, \lambda, v), z_e \in Z_{end}, v \in \Gamma^*\}$$

Anmerkung

- Bei $L_\lambda(A)$ werden die Endzustände nicht benutzt und
- bei $L_Z(A)$ ist der Kellerinhalt v am Ende egal.
- Statt $L_\lambda(A)$ schreiben wir auch (insb. im Skript) $N(A)$ und statt $L_Z(A)$ auch $L(A)$.
- Da das Modell nichtdeterministisch ist, genügt es wie beim NFA, wenn es bei einem Eingabewort **eine Rechnung gibt, die den Keller leert bzw. den Automaten in einen Endzustand bringt**.

Akzeptierte Sprache

$$L_\lambda(A) := \{w \in \Sigma^* \mid (z_0, w, \bot) \vdash^* (z, \lambda, \lambda)\}$$
$$L_Z(A) := \{w \in \Sigma^* \mid (z_0, w, \bot) \vdash^* (z_e, \lambda, v), z_e \in Z_{end}, v \in \Gamma^*\}$$

Anmerkung

- Bei $L_\lambda(A)$ werden die Endzustände nicht benutzt und
- bei $L_Z(A)$ ist der Kellerinhalt v am Ende egal.
- Statt $L_\lambda(A)$ schreiben wir auch (insb. im Skript) $N(A)$ und statt $L_Z(A)$ auch $L(A)$.
- Da das Modell nichtdeterministisch ist, genügt es wie beim NFA, wenn es bei einem Eingabewort **eine Rechnung gibt, die den Keller leert bzw. den Automaten in einen Endzustand bringt**.

Akzeptierte Sprache

$$L_\lambda(A) := \{w \in \Sigma^* \mid (z_0, w, \bot) \vdash^* (z, \lambda, \lambda)\}$$
$$L_Z(A) := \{w \in \Sigma^* \mid (z_0, w, \bot) \vdash^* (z_e, \lambda, v), z_e \in Z_{end}, v \in \Gamma^*\}$$

Anmerkung

- Bei $L_\lambda(A)$ werden die Endzustände nicht benutzt und
- bei $L_Z(A)$ ist der Kellerinhalt v am Ende egal.
- Statt $L_\lambda(A)$ schreiben wir auch (insb. im Skript) $N(A)$ und statt $L_Z(A)$ auch $L(A)$.
- Da das Modell nichtdeterministisch ist, genügt es wie beim NFA, wenn es bei einem Eingabewort **eine Rechnung gibt, die den Keller leert bzw. den Automaten in einen Endzustand bringt**.

Zur Akzeptanzbedingung

Bemerkung

Man kann zeigen, dass beide Akzeptanzbedingungen äquivalent sind, dass also

- Zu einem PDA A und einer Sprache M mit $M = L_Z(A)(= L(A))$ auch ein PDA A' konstruiert werden kann mit $M = L_\lambda(A')(= N(A'))$.
- Zu einem PDA A' und einer Sprache M mit $M = L_\lambda(A')(= N(A'))$ auch ein PDA A konstruiert werden kann mit $M = L_Z(A)(= L(A))$.

Wir wollen dies nicht im Detail beweisen und konzentrieren und meist auf die Akzeptanz mit leerem Keller.

Literaturhinweis

Interessierte finden die Konstruktion(en) im Skript und in [HMU]. Die Kernidee ist bei beiden Richtungen mit einem "zweiten" Kellerbodenzeichen zu arbeiten und dies unter das erste zu schieben. Man kann dann genau feststellen, wann der Keller in einem Automaten leer ist, kann aber trotzdem noch weiter arbeiten.

Zur Akzeptanzbedingung

Bemerkung

Man kann zeigen, dass beide Akzeptanzbedingungen äquivalent sind, dass also

- Zu einem PDA A und einer Sprache M mit $M = L_Z(A)(= L(A))$ auch ein PDA A' konstruiert werden kann mit $M = L_\lambda(A')(= N(A'))$.
- Zu einem PDA A' und einer Sprache M mit $M = L_\lambda(A')(= N(A'))$ auch ein PDA A konstruiert werden kann mit $M = L_Z(A)(= L(A))$.

Wir wollen dies nicht im Detail beweisen und konzentrieren und meist auf die Akzeptanz mit leerem Keller.

Literaturhinweis

Interessierte finden die Konstruktion(en) im Skript und in [HMU]. Die Kernidee ist bei beiden Richtungen mit einem "zweiten" Kellerbodenzeichen zu arbeiten und dies unter das erste zu schieben. Man kann dann genau feststellen, wann der Keller in einem Automaten leer ist, kann aber trotzdem noch weiter arbeiten.

Zur Akzeptanzbedingung

Bemerkung

Man kann zeigen, dass beide Akzeptanzbedingungen äquivalent sind, dass also

- Zu einem PDA A und einer Sprache M mit $M = L_Z(A)(= L(A))$ auch ein PDA A' konstruiert werden kann mit $M = L_\lambda(A')(= N(A'))$.
- Zu einem PDA A' und einer Sprache M mit $M = L_\lambda(A')(= N(A'))$ auch ein PDA A konstruiert werden kann mit $M = L_Z(A)(= L(A))$.

Wir wollen dies nicht im Detail beweisen und konzentrieren und meist auf die Akzeptanz mit leerem Keller.

Literaturhinweis

Interessierte finden die Konstruktion(en) im Skript und in [HMU]. Die Kernidee ist bei beiden Richtungen mit einem "zweiten" Kellerbodenzeichen zu arbeiten und dies unter das erste zu schieben. Man kann dann genau feststellen, wann der Keller in einem Automaten leer ist, kann aber trotzdem noch weiter arbeiten.

Kellerautomat - Funktionsweise

Die (informale) Funktionsweise eines PDA:

- Ein PDA beginnt im Startzustand z_0 und mit $\perp$ im Keller.
- Ist der Automat
  - im Zustand z,
  - liest vom Eingabeband das a
  - und vom Keller das X (dies ist ganz oben auf dem Keller),
- so kann ein Element $(z', w) \in \delta(z, a, X)$ gewählt werden.
- Es wird dann
  - in den Zustand z' gewechselt,
  - der Lesekopf auf dem Eingabeband ein Feld nach rechts bewegt,
  - das X vom Keller gelöscht
  - und dieses X durch w ersetzt (wobei das erste Symbol von w nun ganz oben auf dem Keller ist).

Kellerautomat - Funktionsweise

Die (informale) Funktionsweise eines PDA:

- Ein PDA beginnt im Startzustand z_0 und mit $\perp$ im Keller.
- Ist der Automat
  - im Zustand z,
  - liest vom Eingabeband das a
  - und vom Keller das X (dies ist ganz oben auf dem Keller),
- so kann ein Element $(z', w) \in \delta(z, a, X)$ gewählt werden.
- Es wird dann
  - in den Zustand z' gewechselt,
  - der Lesekopf auf dem Eingabeband ein Feld nach rechts bewegt,
  - das X vom Keller gelöscht
  - und dieses X durch w ersetzt (wobei das erste Symbol von w nun ganz oben auf dem Keller ist).

Kellerautomat - Funktionsweise

Die (informale) Funktionsweise eines PDA:

- Ein PDA beginnt im Startzustand z_0 und mit $\perp$ im Keller.
- Ist der Automat
  - im Zustand z,
  - liest vom Eingabeband das a
  - und vom Keller das X (dies ist ganz oben auf dem Keller),
- so kann ein Element $(z', w) \in \delta(z, a, X)$ gewählt werden.
- Es wird dann
  - in den Zustand z' gewechselt,
  - der Lesekopf auf dem Eingabeband ein Feld nach rechts bewegt,
  - das X vom Keller gelöscht
  - und dieses X durch w ersetzt (wobei das erste Symbol von w nun ganz oben auf dem Keller ist).

Kellerautomat - Funktionsweise

Die (informale) Funktionsweise eines PDA:

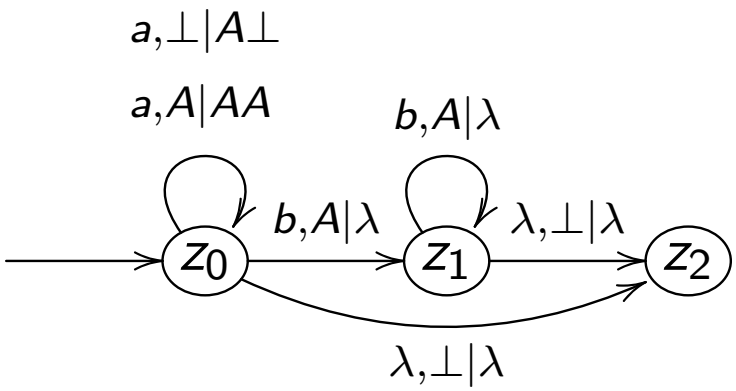
- Ein PDA beginnt im Startzustand z_0 und mit $\perp$ im Keller.
- Ist der Automat
  - im Zustand z,
  - liest vom Eingabeband das a
  - und vom Keller das X (dies ist ganz oben auf dem Keller),
- so kann ein Element $(z', w) \in \delta(z, a, X)$ gewählt werden.
- Es wird dann
  - in den Zustand z' gewechselt,
  - der Lesekopf auf dem Eingabeband ein Feld nach rechts bewegt,
  - das X vom Keller gelöscht
  - und dieses X durch w ersetzt (wobei das erste Symbol von w nun ganz oben auf dem Keller ist).

Beispiel $a^n b^n$

$$L = \{a^n b^n \mid n \in \mathbb{N}\}$$

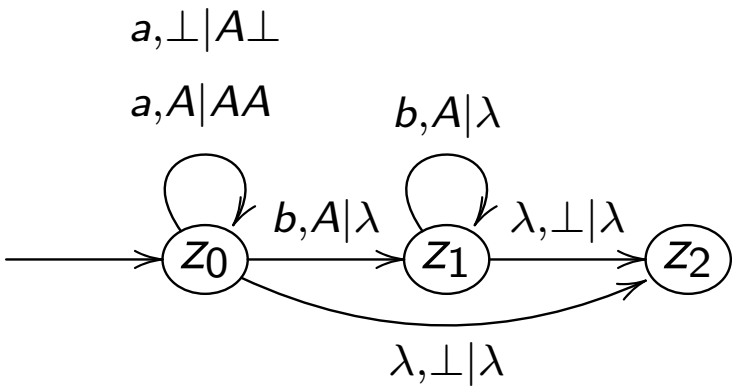
Beispiel $a^n b^n$

$$L = \{a^n b^n \mid n \in \mathbb{N}\}$$



Beispiel $a^n b^n$

$$L = \{a^n b^n \mid n \in \mathbb{N}\}$$

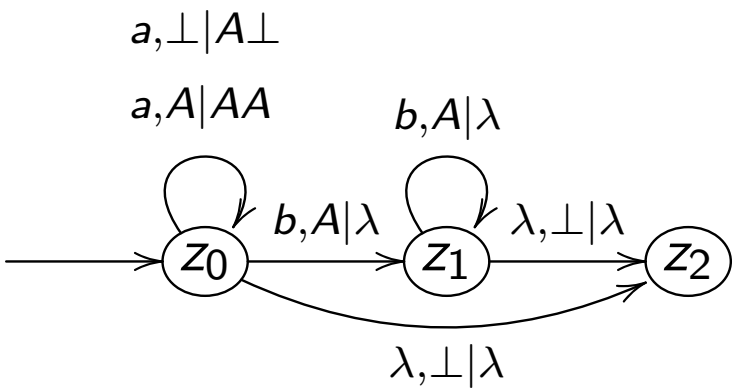


$$(z_0, aabb, \bot) \vdash (z_0, abb, A\bot) \vdash (z_0, bb, AA\bot) \vdash (z_1, b, A\bot) \vdash (z_1, \lambda, \bot) \vdash (z_2, \lambda, \lambda)$$

Akzeptieren! Wort zu Ende gelesen und Keller leer!

Beispiel $a^n b^n$

$$L = \{a^n b^n \mid n \in \mathbb{N}\}$$

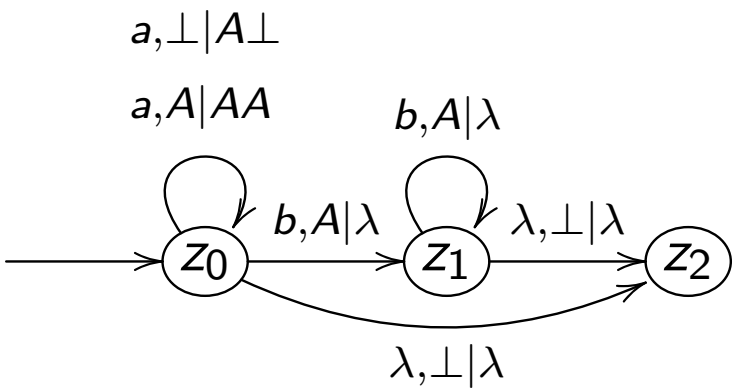


$$(z_0, aabb, \bot) \vdash (z_0, abb, A\bot) \vdash (z_0, bb, AA\bot) \vdash (z_1, b, A\bot) \vdash (z_1, \lambda, \bot) \vdash (z_2, \lambda, \lambda)$$

Akzeptieren! Wort zu Ende gelesen und Keller leer!

Beispiel $a^n b^n$

$$L = \{a^n b^n \mid n \in \mathbb{N}\}$$

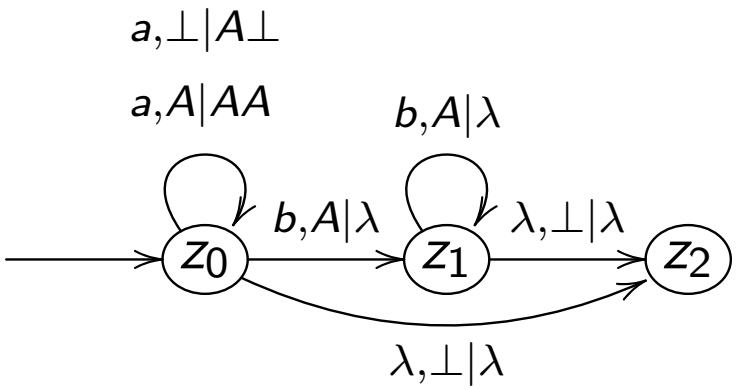


$$(z_0, aabb, \bot) \vdash (z_0, abb, A\bot) \vdash (z_0, bb, AA\bot) \vdash$$
$$(z_1, b, A\bot) \vdash (z_1, \lambda, \bot) \vdash (z_2, \lambda, \lambda)$$

Akzeptieren! Wort zu Ende gelesen und Keller leer!

Beispiel $a^n b^n$

$$L = \{a^n b^n \mid n \in \mathbb{N}\}$$

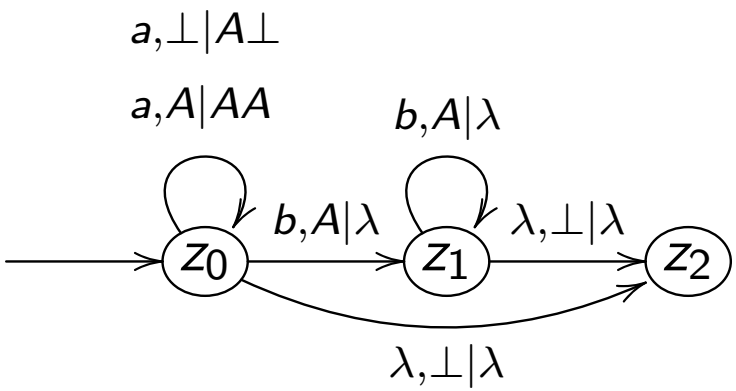


$$(z_0, aabb, \bot) \vdash (z_0, abb, A\bot) \vdash (z_0, bb, AA\bot) \vdash (z_1, b, A\bot) \vdash (z_1, \lambda, \bot) \vdash (z_2, \lambda, \lambda)$$

Akzeptieren! Wort zu Ende gelesen und Keller leer!

Beispiel $a^n b^n$

$$L = \{a^n b^n \mid n \in \mathbb{N}\}$$

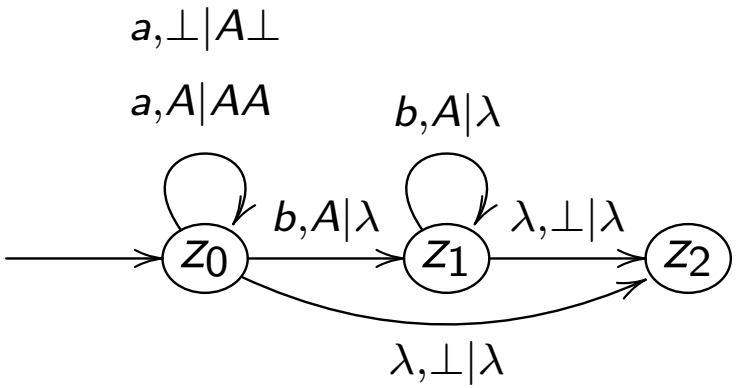


$$(z_0, aabb, \bot) \vdash (z_0, abb, A\bot) \vdash (z_0, bb, AA\bot) \vdash (z_1, b, A\bot) \vdash (z_1, \lambda, \bot) \vdash (z_2, \lambda, \lambda)$$

Akzeptieren! Wort zu Ende gelesen und Keller leer!

Beispiel $a^n b^n$

$$L = \{a^n b^n \mid n \in \mathbb{N}\}$$

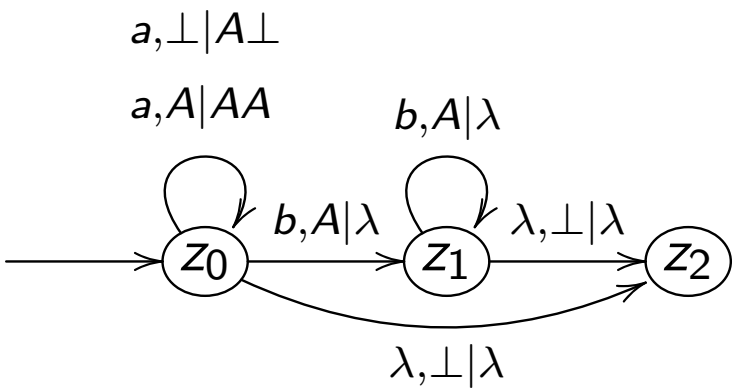


$$(z_0, aabb, \bot) \vdash (z_0, abb, A\bot) \vdash (z_0, bb, AA\bot) \vdash (z_1, b, A\bot) \vdash (z_1, \lambda, \bot) \vdash (z_2, \lambda, \lambda)$$

Akzeptieren! Wort zu Ende gelesen und Keller leer!

Beispiel $a^n b^n$

$$L = \{a^n b^n \mid n \in \mathbb{N}\}$$

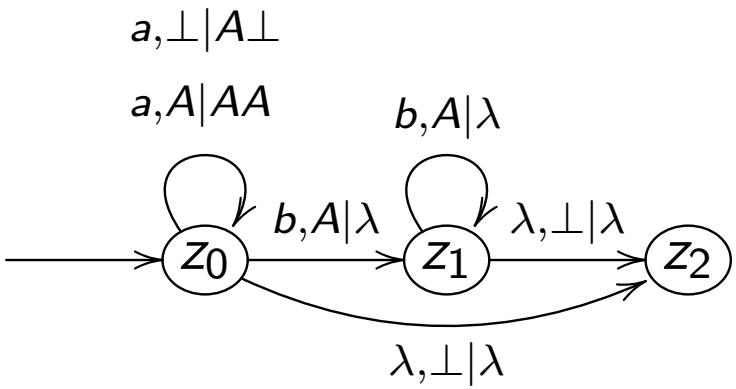


$$(z_0, aabb, \bot) \vdash (z_0, abb, A\bot) \vdash (z_0, bb, AA\bot) \vdash (z_1, b, A\bot) \vdash (z_1, \lambda, \bot) \vdash (z_2, \lambda, \lambda)$$

Akzeptieren! Wort zu Ende gelesen und Keller leer!

Beispiel $a^n b^n$

$$L = \{a^n b^n \mid n \in \mathbb{N}\}$$

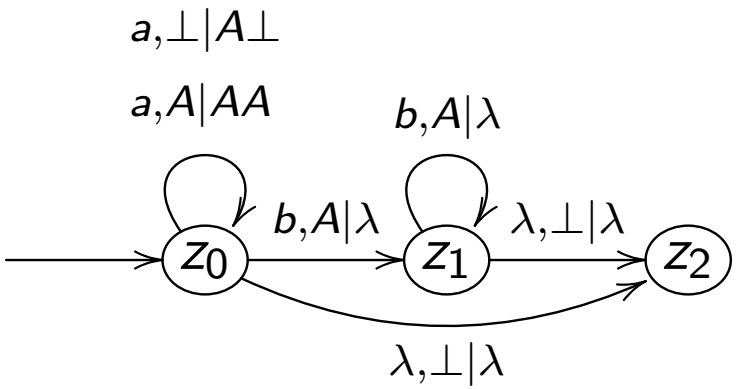


$(z_0, abb, \bot) \vdash (z_0, bb, A\bot) \vdash (z_1, b, \bot) \vdash (z_1, b, \lambda)$

Nicht akzeptieren, da Wort nicht zu Ende gelesen!

Beispiel $a^n b^n$

$$L = \{a^n b^n \mid n \in \mathbb{N}\}$$

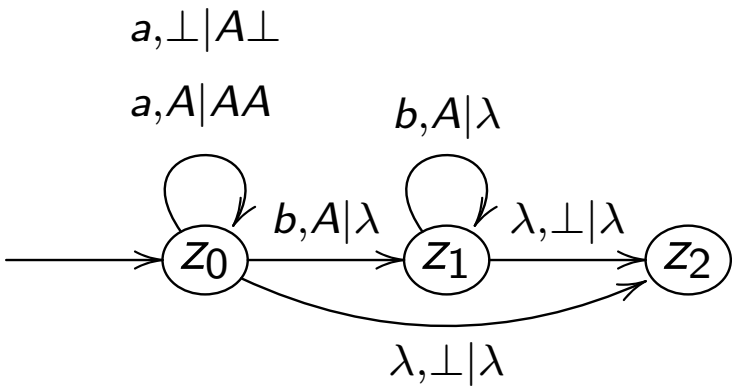


$$(z_0, abb, \bot) \vdash (z_0, bb, A\bot) \vdash (z_1, b, \bot) \vdash (z_1, b, \lambda)$$

Nicht akzeptieren, da Wort nicht zu Ende gelesen!

Beispiel $a^n b^n$

$$L = \{a^n b^n \mid n \in \mathbb{N}\}$$

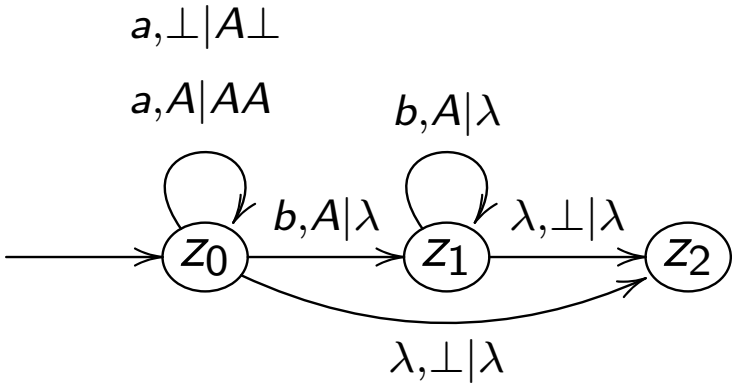


$$(z_0, abb, \bot) \vdash (z_0, bb, A\bot) \vdash (z_1, b, \bot) \vdash (z_1, b, \lambda)$$

Nicht akzeptieren, da Wort nicht zu Ende gelesen!

Beispiel $a^n b^n$

$$L = \{a^n b^n \mid n \in \mathbb{N}\}$$

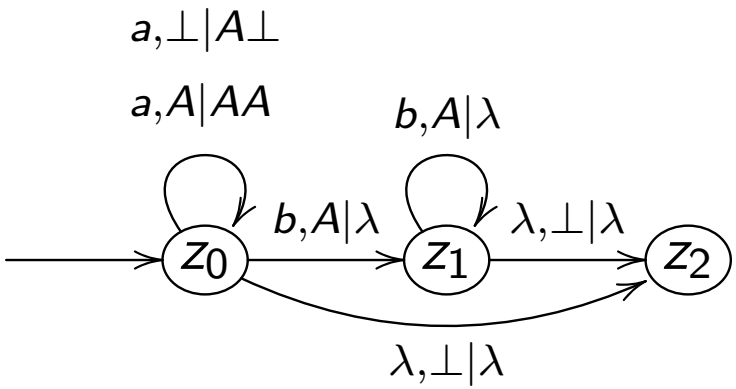


$$(z_0, abb, \bot) \vdash (z_0, bb, A\bot) \vdash (z_1, b, \bot) \vdash (z_1, b, \lambda)$$

Nicht akzeptieren, da Wort nicht zu Ende gelesen!

Beispiel $a^n b^n$

$$L = \{a^n b^n \mid n \in \mathbb{N}\}$$

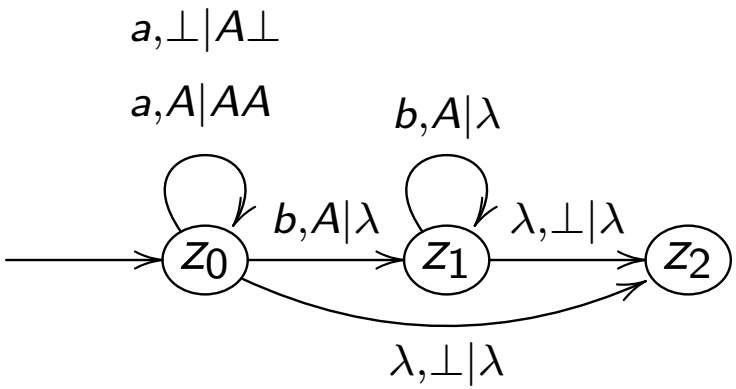


$$(z_0, aa, \bot) \vdash (z_0, a, A\bot) \vdash (z_0, \lambda, AA\bot)$$

Nicht akzeptieren, da Keller nicht leer!

Beispiel $a^n b^n$

$$L = \{a^n b^n \mid n \in \mathbb{N}\}$$

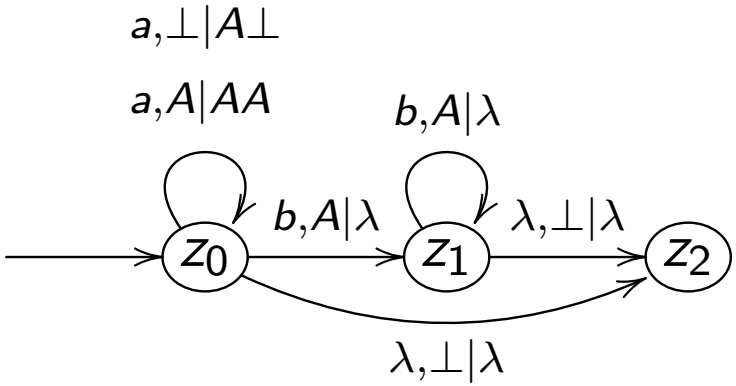


$$(z_0, aa, \bot) \vdash (z_0, a, A\bot) \vdash (z_0, \lambda, AA\bot)$$

Nicht akzeptieren, da Keller nicht leer!

Beispiel $a^n b^n$

$$L = \{a^n b^n \mid n \in \mathbb{N}\}$$



$$(z_0, aa, \bot) \vdash (z_0, a, A\bot) \vdash (z_0, \lambda, AA\bot)$$

Nicht akzeptieren, da Keller nicht leer!

DFA vs. PDA

Wichtige Anmerkung

Alles was ein DFA kann, kann auch ein PDA! Jede Kante (z, a, z') im DFA wird im PDA zur Kante $(z, a, \bot, \bot, z')$. Will man mit leerem Keller akzeptieren, fügt man an jeden Endzustand (des DFAs) z_e noch eine Kante $(z_e, \lambda, \bot, \lambda, z_e)$ hinzu, mit der das $\bot$ gelöscht wird. Der PDA ist also *mindestens so mächtig* wie der DFA. Die Frage ist später noch, ob er tatsächlich mehr kann...

DFA vs. PDA

Wichtige Anmerkung

Alles was ein DFA kann, kann auch ein PDA! Jede Kante (z, a, z') im DFA wird im PDA zur Kante $(z, a, \bot, \bot, z')$. Will man mit leerem Keller akzeptieren, fügt man an jeden Endzustand (des DFAs) z_e noch eine Kante $(z_e, \lambda, \bot, \lambda, z_e)$ hinzu, mit der das $\bot$ gelöscht wird. Der PDA ist also *mindestens so mächtig* wie der DFA. Die Frage ist später noch, ob er tatsächlich mehr kann...

DFA vs. PDA

Wichtige Anmerkung

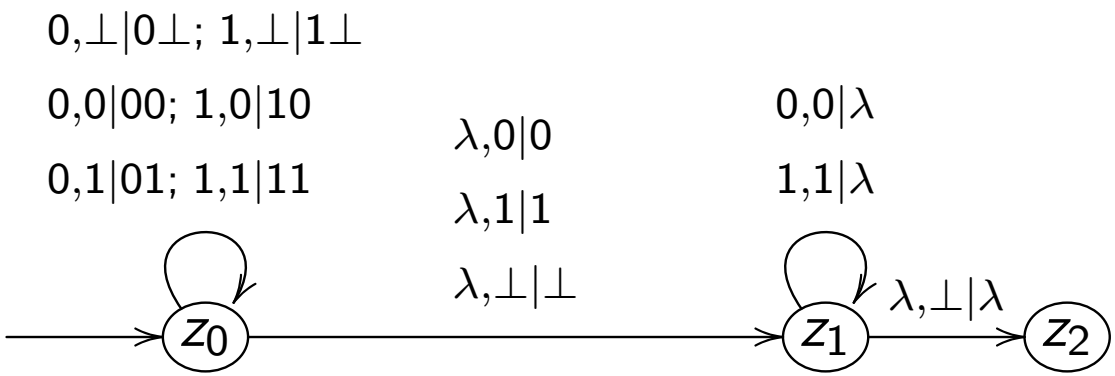
Alles was ein DFA kann, kann auch ein PDA! Jede Kante (z, a, z') im DFA wird im PDA zur Kante $(z, a, \bot, \bot, z')$. Will man mit leerem Keller akzeptieren, fügt man an jeden Endzustand (des DFAs) z_e noch eine Kante $(z_e, \lambda, \bot, \lambda, z_e)$ hinzu, mit der das $\bot$ gelöscht wird. Der PDA ist also *mindestens so mächtig* wie der DFA. Die Frage ist später noch, ob er tatsächlich mehr kann...

DFA vs. PDA

Wichtige Anmerkung

Alles was ein DFA kann, kann auch ein PDA! Jede Kante (z, a, z') im DFA wird im PDA zur Kante $(z, a, \bot, \bot, z')$. Will man mit leerem Keller akzeptieren, fügt man an jeden Endzustand (des DFAs) z_e noch eine Kante $(z_e, \lambda, \bot, \lambda, z_e)$ hinzu, mit der das $\bot$ gelöscht wird. Der PDA ist also *mindestens so mächtig* wie der DFA. Die Frage ist später noch, ob er tatsächlich mehr kann...

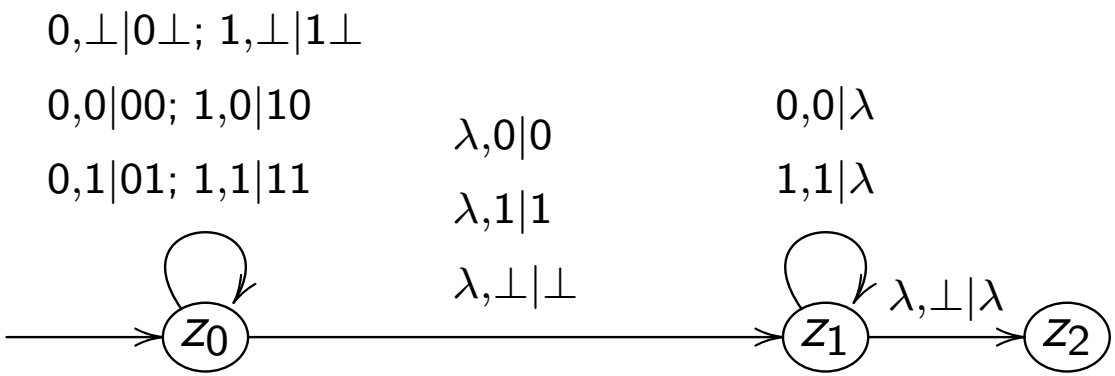
Fragen ...



In welchen Zuständen kann man nach Lesen von 0011 sein?

1. Nur z_0
2. z_0 und z_1
3. z_0, z_1 und z_2
4. Nur z_2

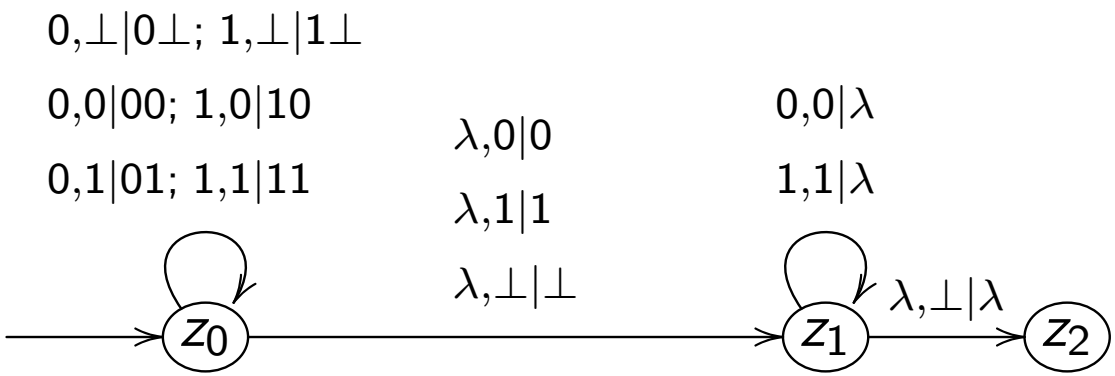
Fragen ...



Unabhängig vom Eingabewort: In welchen Zuständen kann man sein, so dass der Keller leer ist?

1. Nur z_0
2. Nur z_1
3. Nur z_2
4. In z_1 und z_2 kann der Keller leer sein

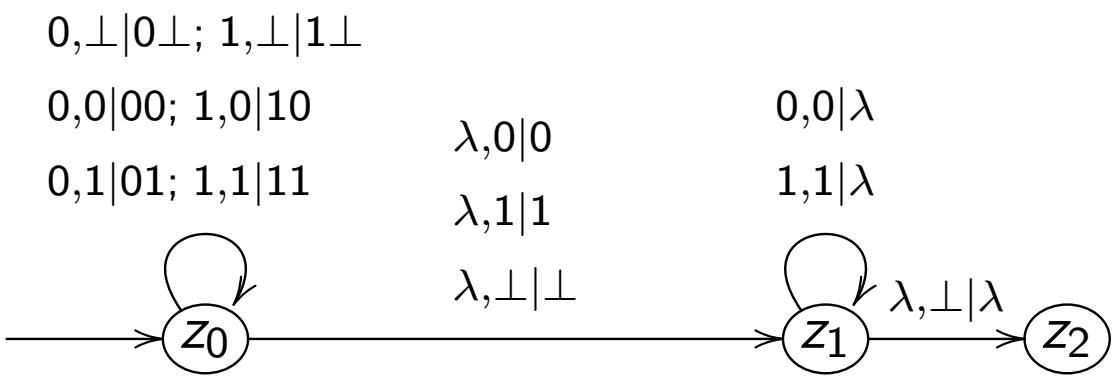
Fragen ...



In welchen Zuständen kann man nach Lesen von 0110 sein?

1. Nur z_0
2. z_0 und z_1
3. z_0, z_1 und z_2
4. Nur z_2

Fragen ...



Bei diesem Automaten kann man mit leeren Keller nicht mehr lesen, weil es keine Kante aus z_2 heraus gibt. Ist es im Allgemeinen möglich bei leerem Keller weitere Aktionen auszuführen? Also z.B. hier eine Kante $(z_2, \lambda, \lambda, \lambda, z_0)$ einzufügen?

1. Ja!
2. Nein!

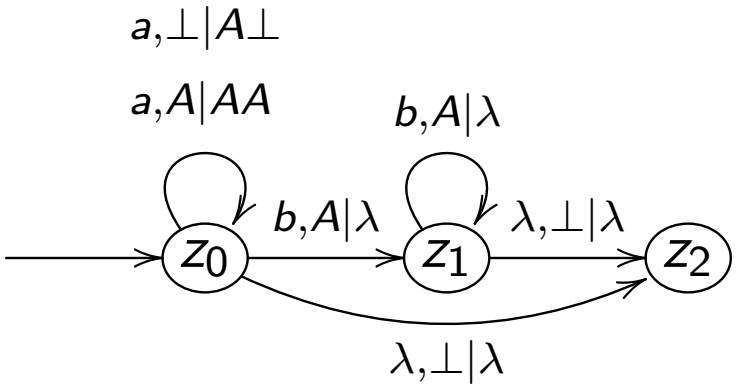
Zur Nachbereitung

Zur Nachbereitung

Richtige Antworten sind:

1. 2
2. 3
3. 3
4. 2

Beispiel $a^n b^n$

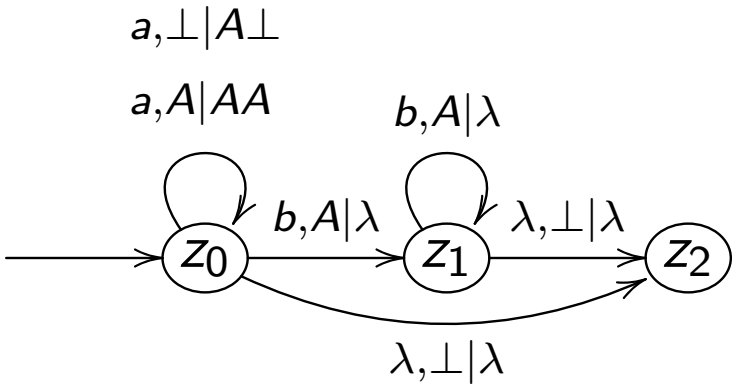


Behauptung

$$N(A) = \{a^n b^n \mid n \in \mathbb{N}\} =: M$$

Vorüberlegung: Es gilt $\lambda \in M$ (mit $n = 0$) und mit $(z_0, \lambda, \bot) \vdash (z_2, \lambda, \lambda)$ auch $\lambda \in N(A)$. Wir können daher nachfolgend davon ausgehen, dass $|w| > 0$ gilt.

Beispiel $a^n b^n$

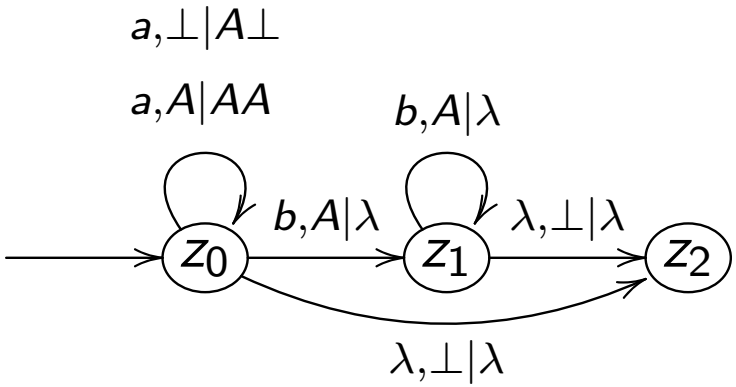


Behauptung

$$N(A) = \{a^n b^n \mid n \in \mathbb{N}\} =: M$$

Vorüberlegung: Es gilt $\lambda \in M$ (mit $n = 0$) und mit $(z_0, \lambda, \bot) \vdash (z_2, \lambda, \lambda)$ auch $\lambda \in N(A)$. Wir können daher nachfolgend davon ausgehen, dass $|w| > 0$ gilt.

Korrektheitsbeweis

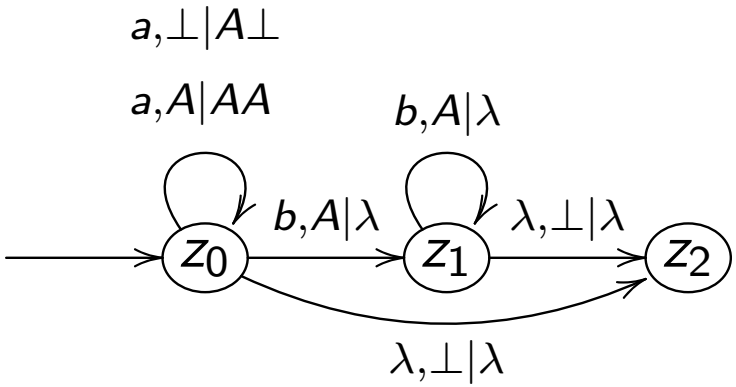


$M := \{a^n b^n \mid n \in \mathbb{N}\} \subseteq N(A)$. Sei $w = a^n b^n \in M$ für ein $n \geq 1$. In A gilt nun

$$\begin{aligned}(z_0, a^n b^n, \bot) &\vdash (z_0, a^{n-1} b^n, A\bot) \vdash \ldots \vdash (z_0, b^n, A^n \bot) \\ &\vdash (z_1, b^{n-1}, A^{n-1} \bot) \vdash \ldots \vdash (z_1, \lambda, \bot) \\ &\vdash (z_2, \lambda, \lambda)\end{aligned}$$

und folglich auch $w \in N(A)$.

Korrektheitsbeweis

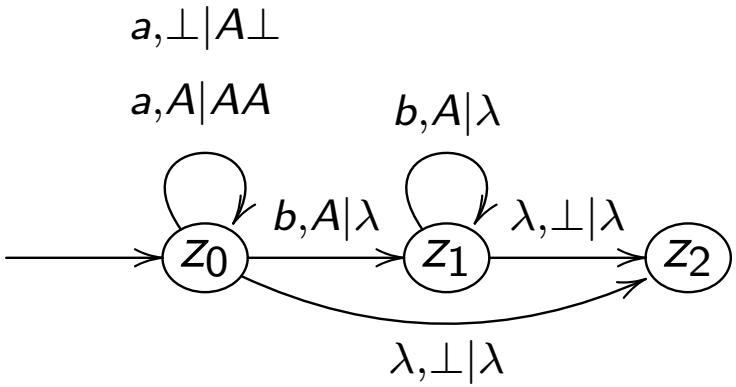


$M := \{a^n b^n \mid n \in \mathbb{N}\} \subseteq N(A)$. Sei $w = a^n b^n \in M$ für ein $n \geq 1$. In A gilt nun

$$\begin{aligned}(z_0, a^n b^n, \bot) &\vdash (z_0, a^{n-1} b^n, A\bot) \vdash \ldots \vdash (z_0, b^n, A^n \bot) \\ &\vdash (z_1, b^{n-1}, A^{n-1} \bot) \vdash \ldots \vdash (z_1, \lambda, \bot) \\ &\vdash (z_2, \lambda, \lambda)\end{aligned}$$

und folglich auch $w \in N(A)$.

Korrektheitsbeweis

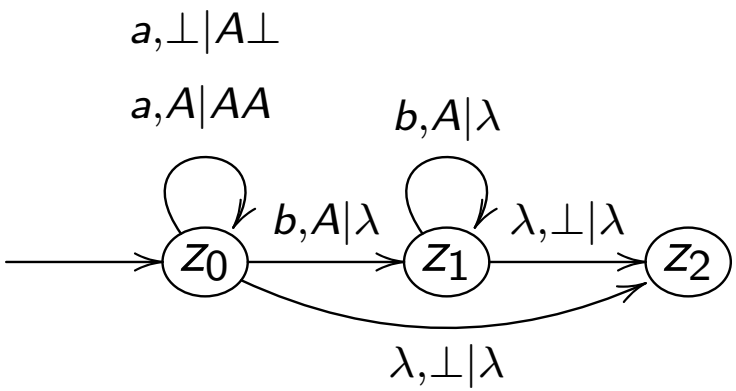


$M := \{a^n b^n \mid n \in \mathbb{N}\} \subseteq N(A)$. Sei $w = a^n b^n \in M$ für ein $n \geq 1$. In A gilt nun

$$\begin{aligned}(z_0, a^n b^n, \bot) &\vdash (z_0, a^{n-1} b^n, A\bot) \vdash \ldots \vdash (z_0, b^n, A^n \bot) \\ &\vdash (z_1, b^{n-1}, A^{n-1} \bot) \vdash \ldots \vdash (z_1, \lambda, \bot) \\ &\vdash (z_2, \lambda, \lambda)\end{aligned}$$

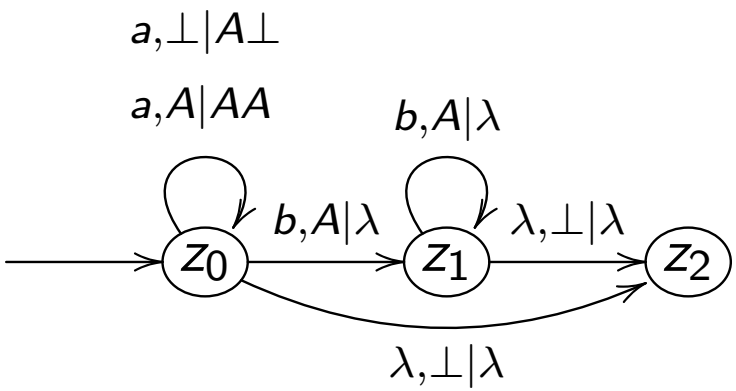
und folglich auch $w \in N(A)$.

Korrektheitsbeweis



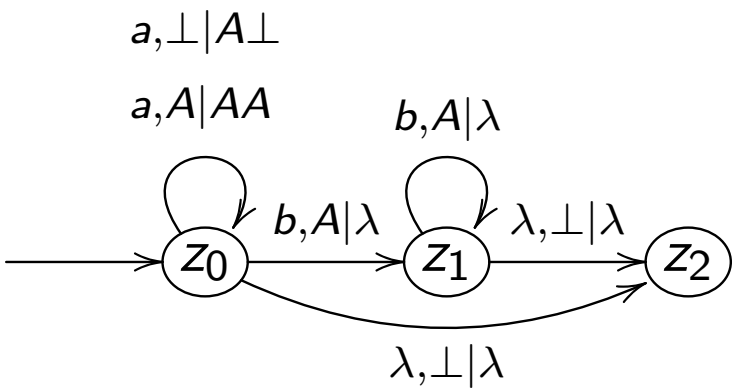
$N(A) \subseteq \{a^n b^n \mid n \in \mathbb{N}\} = M$. Sei $w \in N(A)$ mit $|w| \geq 1$. Der Kantenübergang von z_0 nach z_2 ist nur ganz zu Anfang möglich, da sonst in z_0 nie vom Keller gelöscht wird und das Kellerbodensymbol dann nicht mehr erreichbar ist. D.h. diese Kante kann nur genutzt werden, um das leere Wort zu akzeptieren und ist hier nicht von Bedeutung.

Korrektheitsbeweis



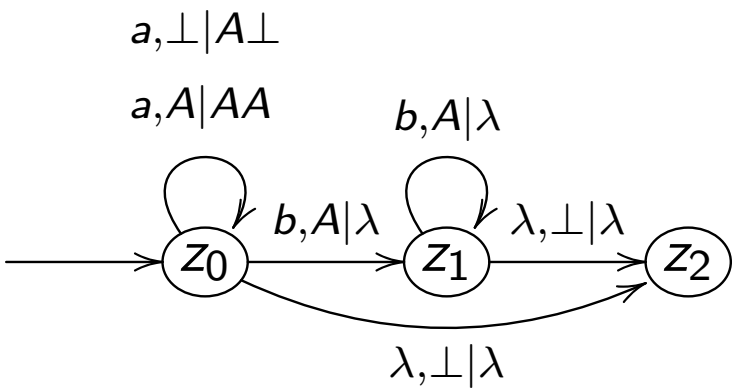
$N(A) \subseteq \{a^n b^n \mid n \in \mathbb{N}\} = M$. Sei $w \in N(A)$ mit $|w| \geq 1$. Der Kantenübergang von z_0 nach z_2 ist nur ganz zu Anfang möglich, da sonst in z_0 nie vom Keller gelöscht wird und das Kellerbodensymbol dann nicht mehr erreichbar ist. D.h. diese Kante kann nur genutzt werden, um das leere Wort zu akzeptieren und ist hier nicht von Bedeutung.

Korrektheitsbeweis



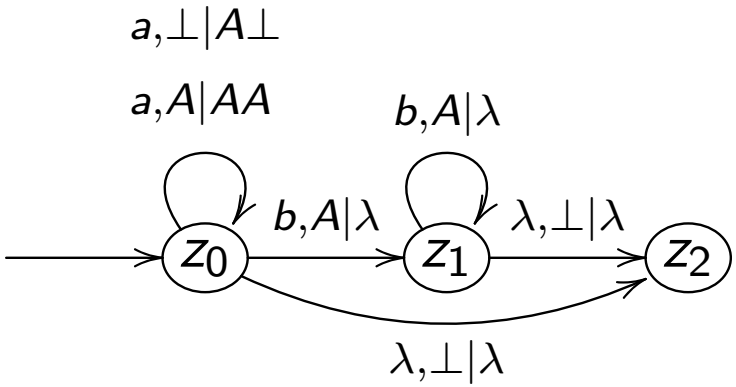
$N(A) \subseteq \{a^n b^n \mid n \in \mathbb{N}\} = M$. Sei $w \in N(A)$ mit $|w| \geq 1$. Der Kantenübergang von z_0 nach z_2 ist nur ganz zu Anfang möglich, da sonst in z_0 nie vom Keller gelöscht wird und das Kellerbodensymbol dann nicht mehr erreichbar ist. D.h. diese Kante kann nur genutzt werden, um das leere Wort zu akzeptieren und ist hier nicht von Bedeutung.

Korrektheitsbeweis



$N(A) \subseteq \{a^n b^n \mid n \in \mathbb{N}\} = M$. Sei $w \in N(A)$ mit $|w| \geq 1$. Der Kantenübergang von z_0 nach z_2 ist nur ganz zu Anfang möglich, da sonst in z_0 nie vom Keller gelöscht wird und das Kellerbodensymbol dann nicht mehr erreichbar ist. D.h. diese Kante kann nur genutzt werden, um das leere Wort zu akzeptieren und ist hier nicht von Bedeutung.

Korrektheitsbeweis

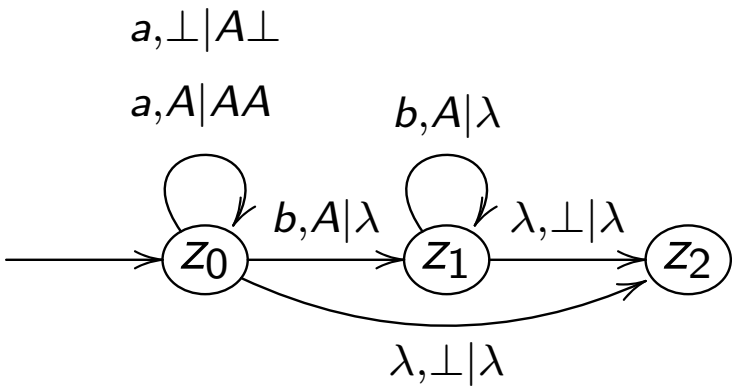


An den Kantenübergängen erkennt man nun, dass

- ausschließlich in z_0 as gelesen werden können.
- Für jedes gelesene a wird zudem ein A auf den Keller geschrieben.
- Zudem muss w mit mindestens einem a beginnen, da sonst kein Übergang möglich ist.

Insgesamt heißt dies, dass w mit a^i für ein $i \geq 1$ beginnen muss.

Korrektheitsbeweis

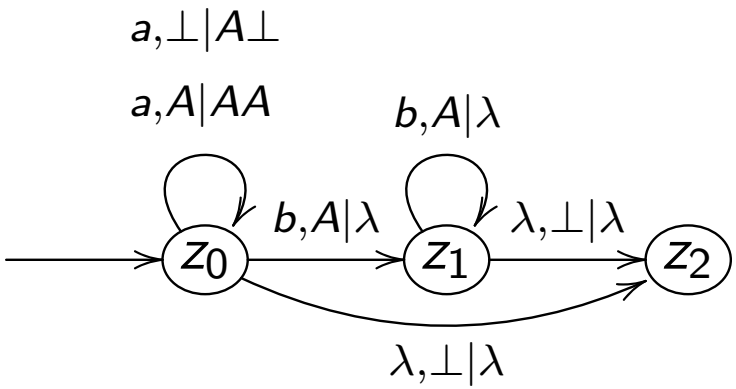


An den Kantenübergängen erkennt man nun, dass

- ausschließlich in z_0 as gelesen werden können.
- Für jedes gelesene a wird zudem ein A auf den Keller geschrieben.
- Zudem muss w mit mindestens einem a beginnen, da sonst kein Übergang möglich ist.

Insgesamt heißt dies, dass w mit a^i für ein $i \geq 1$ beginnen muss.

Korrektheitsbeweis

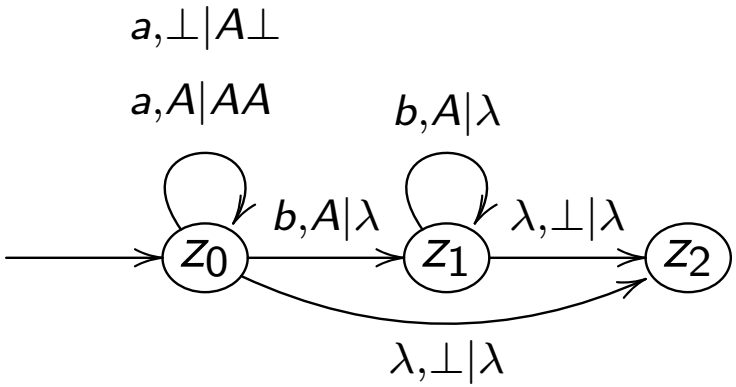


An den Kantenübergängen erkennt man nun, dass

- ausschließlich in z_0 as gelesen werden können.
- Für jedes gelesene a wird zudem ein A auf den Keller geschrieben.
- Zudem muss w mit mindestens einem a beginnen, da sonst kein Übergang möglich ist.

Insgesamt heißt dies, dass w mit a^i für ein $i \geq 1$ beginnen muss.

Korrektheitsbeweis

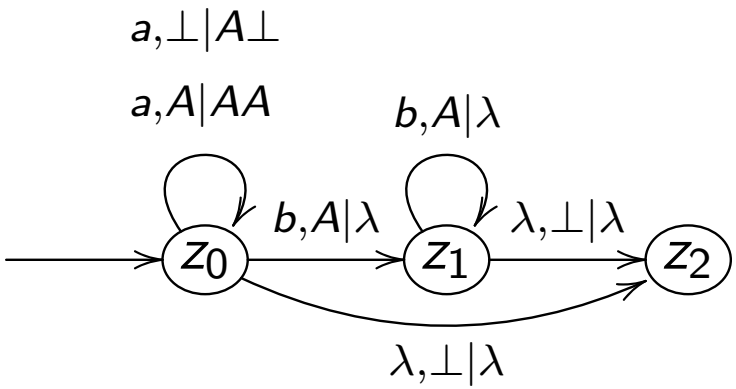


An den Kantenübergängen erkennt man nun, dass

- ausschließlich in z_0 as gelesen werden können.
- Für jedes gelesene a wird zudem ein A auf den Keller geschrieben.
- Zudem muss w mit mindestens einem a beginnen, da sonst kein Übergang möglich ist.

Insgesamt heißt dies, dass w mit a^i für ein $i \geq 1$ beginnen muss.

Korrektheitsbeweis

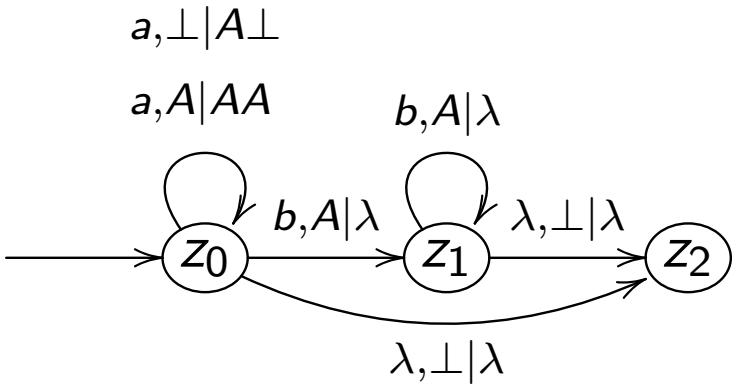


An den Kantenübergängen erkennt man nun, dass

- ausschließlich in z_0 as gelesen werden können.
- Für jedes gelesene a wird zudem ein A auf den Keller geschrieben.
- Zudem muss w mit mindestens einem a beginnen, da sonst kein Übergang möglich ist.

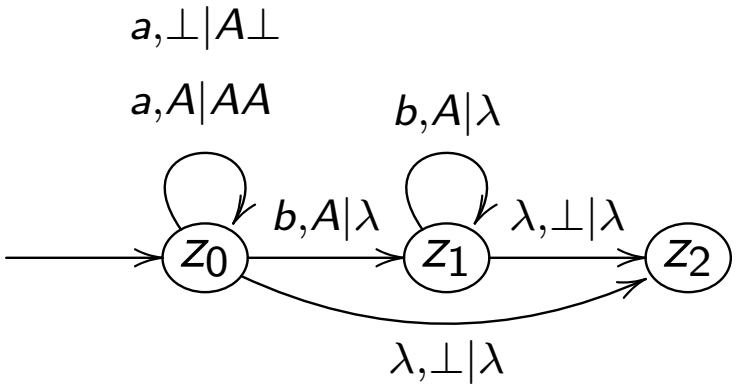
Insgesamt heißt dies, dass w mit a^i für ein $i \geq 1$ beginnen muss.

Korrektheitsbeweis



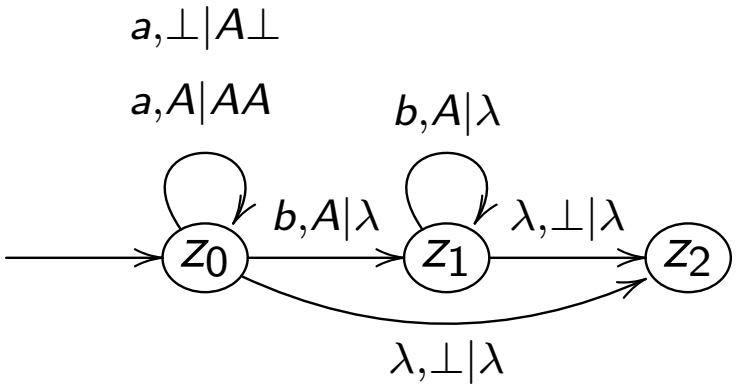
Sei also $w = a^i v$, wobei v das Restwort ist und mit b beginnen muss (da sonst z_0 nicht verlassen werden kann). Nach Lesen von a^i ist A dann in der Konfiguration $(z_0, v, A^i \perp)$. An den Kantenübergängen erkennt man nun, dass genau i mal das b gelesen werden muss, um an das $\perp$ heranzukommen. Dies geschieht beim Übergang nach z_1 und dann in z_1. Bei jedem Lesen eines b wird ein A gelöscht. Da wir an $\perp$ herankommen müssen, ist es also weder möglich weniger noch mehr als i bs zu lesen.

Korrektheitsbeweis



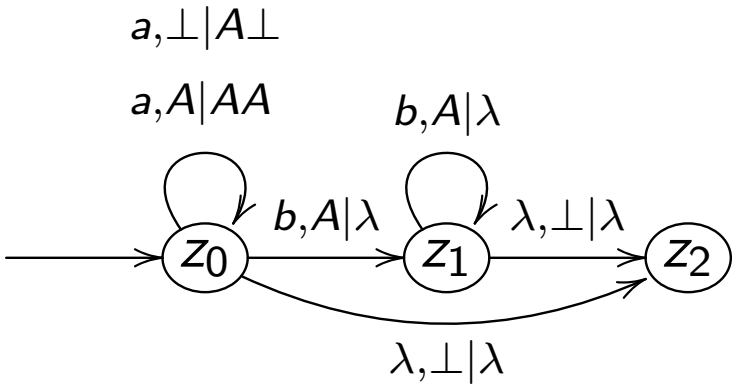
Sei also $w = a^i v$, wobei v das Restwort ist und mit b beginnen muss (da sonst z_0 nicht verlassen werden kann). Nach Lesen von a^i ist A dann in der Konfiguration $(z_0, v, A^i \perp)$. An den Kantenübergängen erkennt man nun, dass genau i mal das b gelesen werden muss, um an das $\perp$ heranzukommen. Dies geschieht beim Übergang nach z_1 und dann in z_1. Bei jedem Lesen eines b wird ein A gelöscht. Da wir an $\perp$ herankommen müssen, ist es also weder möglich weniger noch mehr als i bs zu lesen.

Korrektheitsbeweis



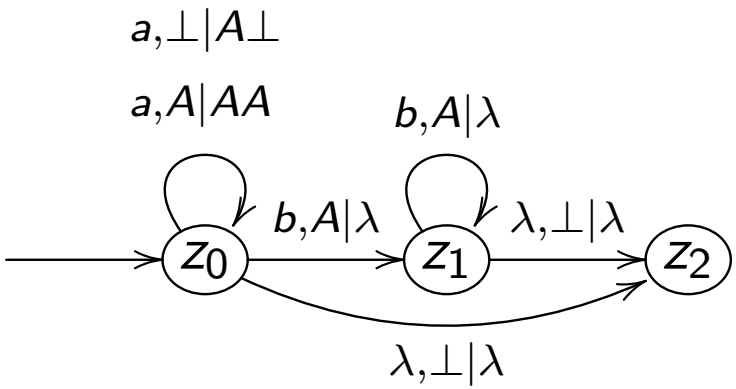
Sei also $w = a^i v$, wobei v das Restwort ist und mit b beginnen muss (da sonst z_0 nicht verlassen werden kann). Nach Lesen von a^i ist A dann in der Konfiguration $(z_0, v, A^i \perp)$. An den Kantenübergängen erkennt man nun, dass genau i mal das b gelesen werden muss, um an das $\perp$ heranzukommen. Dies geschieht beim Übergang nach z_1 und dann in z_1. Bei jedem Lesen eines b wird ein A gelöscht. Da wir an $\perp$ herankommen müssen, ist es also weder möglich weniger noch mehr als i bs zu lesen.

Korrektheitsbeweis



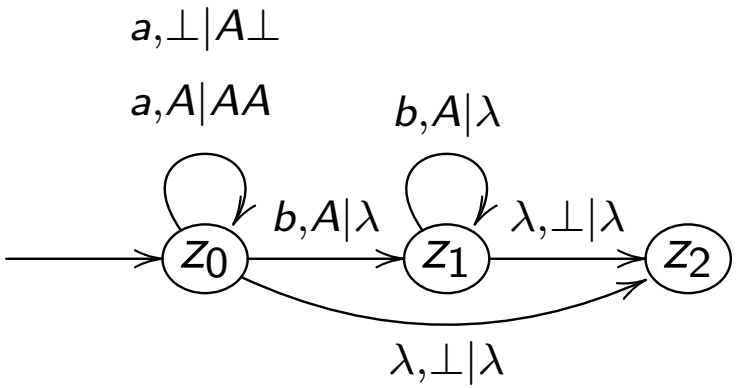
Sei also $w = a^i v$, wobei v das Restwort ist und mit b beginnen muss (da sonst z_0 nicht verlassen werden kann). Nach Lesen von a^i ist A dann in der Konfiguration $(z_0, v, A^i \bot)$. An den Kantenübergängen erkennt man nun, dass genau i mal das b gelesen werden muss, um an das $\bot$ heranzukommen. Dies geschieht beim Übergang nach z_1 und dann in z_1. Bei jedem Lesen eines b wird ein A gelöscht. Da wir an $\bot$ herankommen müssen, ist es also weder möglich weniger noch mehr als i bs zu lesen.

Korrektheitsbeweis



Es muss also gerade $v = b^i$ sein. Erst nun ist das $\perp$ im Keller erreichbar und wir können es beim Übergang nach z_2 löschen. Damit hat w die Form $a^i b^i$ für ein $i \geq 1$ und damit gilt auch $w \in M$.

Korrektheitsbeweis



Es muss also gerade $v = b^i$ sein. Erst nun ist das $\perp$ im Keller erreichbar und wir können es beim Übergang nach z_2 löschen. Damit hat w die Form $a^i b^i$ für ein $i \geq 1$ und damit gilt auch $w \in M$.

Die Technik

Zur Zähltechnik

Diese Technik des Zählens kann häufig benutzt werden. Varianten:

- Um z.B. $a^n b^{2n}$ zu akzeptieren, achtet man beim Löschen der A im Keller darauf, dass hierfür immer zwei b gelesen werden müssen (oder alternativ: schreibt stets zwei A beim Lesen eines a).
- Für $a^n b^m$ mit $n > m$ muss der Keller beim letzten zu lesenden Eingabesymbol nicht leer werden. Er muss dann aber noch in einem Zustand (in dem man davon ausgeht, dass das Wort zu Ende gelesen wurde) noch geleert werden.
- Für $a^n b^m$ mit $n < m$ liest man, wenn man $\perp$ erreicht hat, noch beliebig viele weitere b.

Die Technik

Zur Zähltechnik

Diese Technik des Zählens kann häufig benutzt werden. Varianten:

- Um z.B. $a^n b^{2n}$ zu akzeptieren, achtet man beim Löschen der A im Keller darauf, dass hierfür immer zwei b gelesen werden müssen (oder alternativ: schreibt stets zwei A beim Lesen eines a).
- Für $a^n b^m$ mit $n > m$ muss der Keller beim letzten zu lesenden Eingabesymbol nicht leer werden. Er muss dann aber noch in einem Zustand (in dem man davon ausgeht, dass das Wort zu Ende gelesen wurde) noch geleert werden.
- Für $a^n b^m$ mit $n < m$ liest man, wenn man $\perp$ erreicht hat, noch beliebig viele weitere b.

Die Technik

Zur Zähltechnik

Diese Technik des Zählens kann häufig benutzt werden. Varianten:

- Um z.B. $a^n b^{2n}$ zu akzeptieren, achtet man beim Löschen der A im Keller darauf, dass hierfür immer zwei b gelesen werden müssen (oder alternativ: schreibt stets zwei A beim Lesen eines a).
- Für $a^n b^m$ mit $n > m$ muss der Keller beim letzten zu lesenden Eingabesymbol nicht leer werden. Er muss dann aber noch in einem Zustand (in dem man davon ausgeht, dass das Wort zu Ende gelesen wurde) noch geleert werden.
- Für $a^n b^m$ mit $n < m$ liest man, wenn man $\perp$ erreicht hat, noch beliebig viele weitere b.

Die Technik

Zur Zähltechnik

Diese Technik des Zählens kann häufig benutzt werden. Varianten:

- Um z.B. $a^n b^{2n}$ zu akzeptieren, achtet man beim Löschen der A im Keller darauf, dass hierfür immer zwei b gelesen werden müssen (oder alternativ: schreibt stets zwei A beim Lesen eines a).
- Für $a^n b^m$ mit $n > m$ muss der Keller beim letzten zu lesenden Eingabesymbol nicht leer werden. Er muss dann aber noch in einem Zustand (in dem man davon ausgeht, dass das Wort zu Ende gelesen wurde) noch geleert werden.
- Für $a^n b^m$ mit $n < m$ liest man, wenn man $\perp$ erreicht hat, noch beliebig viele weitere b.

Die Technik

Zur Zähltechnik

Weitere Varianten:

- Will man Worte akzeptieren, bei denen jedes Präfix mehr 0en als 1en hat, kann man auch mit einem Zähler arbeiten, muss diesen aber dynamisch erhöhen und verringern und ggf. abbrechen, wenn man versucht ihn bei $\perp$ noch weiter zu verringern. ($\Rightarrow$ Präsenzaufgabe)
- Will man Worte akzeptieren, die gleich viele 0en wie 1en haben, kann man ähnlich verfahren, aber der Zähler "flippt" sozusagen von einem 0-Zähler zu einem 1-Zähler (oder andersherum) und am Ende des Wortes muss der Zähler leer sein. ($\Rightarrow$ Übungsaufgabe)

Anmerkung

Definition des Präfix auf der nächsten Folie.

Die Technik

Zur Zähltechnik

Weitere Varianten:

- Will man Worte akzeptieren, bei denen jedes Präfix mehr 0en als 1en hat, kann man auch mit einem Zähler arbeiten, muss diesen aber dynamisch erhöhen und verringern und ggf. abbrechen, wenn man versucht ihn bei $\perp$ noch weiter zu verringern. ($\Rightarrow$ Präsenzaufgabe)
- Will man Worte akzeptieren, die gleich viele 0en wie 1en haben, kann man ähnlich verfahren, aber der Zähler "flippt" sozusagen von einem 0-Zähler zu einem 1-Zähler (oder andersherum) und am Ende des Wortes muss der Zähler leer sein. ($\Rightarrow$ Übungsaufgabe)

Anmerkung

Definition des Präfix auf der nächsten Folie.

Zwei Definitionen

Definition (Präfix)

Unter einem **Präfix** eines Wortes $w \in \Sigma^*$ versteht man ein Wort $v \in \Sigma^*$ so dass ein $u \in \Sigma^*$ existiert mit $w = vu$. Informal: ein beliebiges Anfangsstück von w ist ein Präfix. Ist bspw. $w = 1001$, so ist 1 ein Präfix, ebenso wie 10 und 100. Außerdem sind auch λ und w selbst Präfixe von w.

Definition (w^{rev})

Das **Spiegelwort** w^{rev} zu einem Wort $w \in \Sigma^*$ ist definiert durch

1. $\lambda^{rev} = \lambda$ und
2. $(ux)^{rev} = x \cdot u^{rev}$ mit $x \in \Sigma$ und $u \in \Sigma^*$

Bspw. ist für $w = 1011$ dann $w^{rev} = 1101$.
Für Mengen notieren wir $L^{rev} = \{w^{rev} \mid w \in L\}$.

Zwei Definitionen

Definition (Präfix)

Unter einem **Präfix** eines Wortes $w \in \Sigma^*$ versteht man ein Wort $v \in \Sigma^*$ so dass ein $u \in \Sigma^*$ existiert mit $w = vu$. Informal: ein beliebiges Anfangsstück von w ist ein Präfix. Ist bspw. $w = 1001$, so ist 1 ein Präfix, ebenso wie 10 und 100. Außerdem sind auch λ und w selbst Präfixe von w.

Definition (w^{rev})

Das **Spiegelwort** w^{rev} zu einem Wort $w \in \Sigma^*$ ist definiert durch

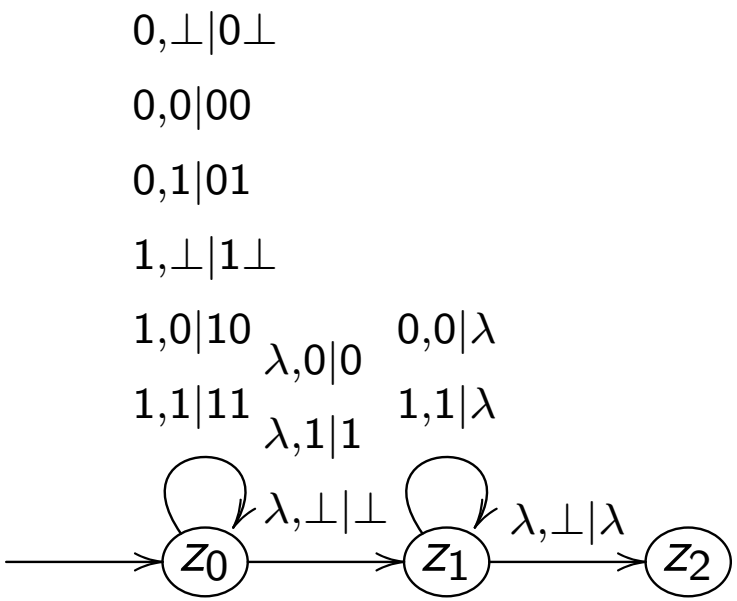
1. $\lambda^{rev} = \lambda$ und
2. $(ux)^{rev} = x \cdot u^{rev}$ mit $x \in \Sigma$ und $u \in \Sigma^*$

Bspw. ist für $w = 1011$ dann $w^{rev} = 1101$.
Für Mengen notieren wir $L^{rev} = \{w^{rev} \mid w \in L\}$.

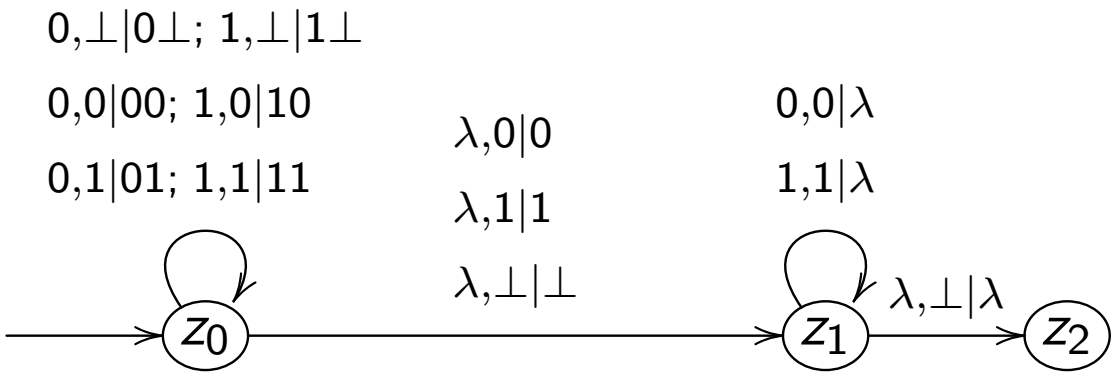
Ein zweites Beispiel: ww^{rev}

Wir wollen nun einen PDA konstruieren für

$$L := \{ww^{rev} \mid w \in \{0,1\}^*\}$$



Korrektheitsbeweis



Die Gültigkeit von $L := \{ww^{rev} \mid w \in \{0,1\}^*\} = N(A)$ zeigen wir nächstes Mal...

Fragen...

Gilt stets für jeden PDA $L(A) = N(A)$?

1. Ja
2. Nein

Fragen...

Kann ein PDA noch weiter lesen/arbeiten, obwohl sein Keller leer ist?

1. Ja
2. Nein

Fragen...

Muss ein PDA stets vom Band lesen (so wie ein DFA)?

1. Ja
2. Nein

Zur Nachbereitung

Zur Nachbereitung

Richtige Antworten sind:

1. 2
2. 2
3. 2

Zusammenfassung

Wir haben

- PDAs / Kellerautomaten kennengelernt
- Wieder die Begriffe wie Konfiguration, Rechnung, akzeptierte Sprache usw. eingeführt

Dabei

- können PDAs mit leerem Keller oder mit Endzustand akzeptieren.
- Beide Akzeptanzbedingungen sind äquivalent (ohne Beweis).

Wir haben

- einen PDA für $L = \{a^n b^n \mid n \in \mathbb{N}\}$ gebaut
  - und dabei den Keller als Zähler benutzt
- einen PDA für $L = \{ww^{rev} \mid w \in \{0,1\}^*\}$ gebaut
  - und dabei den Keller als Speicher benutzt (Details folgen)

Zusammenfassung

Wir haben

- PDAs / Kellerautomaten kennengelernt
- Wieder die Begriffe wie Konfiguration, Rechnung, akzeptierte Sprache usw. eingeführt

Dabei

- können PDAs mit leerem Keller oder mit Endzustand akzeptieren.
- Beide Akzeptanzbedingungen sind äquivalent (ohne Beweis).

Wir haben

- einen PDA für $L = \{a^n b^n \mid n \in \mathbb{N}\}$ gebaut
  - und dabei den Keller als Zähler benutzt
- einen PDA für $L = \{ww^{rev} \mid w \in \{0,1\}^*\}$ gebaut
  - und dabei den Keller als Speicher benutzt (Details folgen)

Zusammenfassung

Wir haben

- PDAs / Kellerautomaten kennengelernt
- Wieder die Begriffe wie Konfiguration, Rechnung, akzeptierte Sprache usw. eingeführt

Dabei

- können PDAs mit leerem Keller oder mit Endzustand akzeptieren.
- Beide Akzeptanzbedingungen sind äquivalent (ohne Beweis).

Wir haben

- einen PDA für $L = \{a^n b^n \mid n \in \mathbb{N}\}$ gebaut
  - und dabei den Keller als Zähler benutzt
- einen PDA für $L = \{ww^{rev} \mid w \in \{0,1\}^*\}$ gebaut
  - und dabei den Keller als Speicher benutzt (Details folgen)