Lösungen zu den Übungsaufgaben im Buch *Automaten und Sprachen: Theoretische Informatik für die Praxis* von Andreas Müller, ISBN 978-3-662-70145-4 (Softcover), ISBN 978-3-662-70146-1 (eBook), https://link.springer.com/book/10.1007/978-3-662-70146-1. Website zum Buch: https://autospr.ch

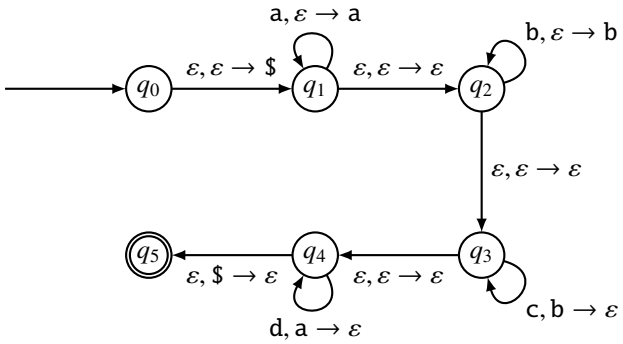
Kapitel 7: Stackautomaten

7.1. Konstruieren Sie einen Stackautomaten, der die Sprache

$$L = \{\mathtt{a}^n\mathtt{b}^m\mathtt{c}^m\mathtt{d}^n \mid m, n \geq 0\}.$$

über dem Alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$ akzeptiert.

Lösung. Der folgende Automat akzeptiert die Sprache L:



○

7.2. Konstruieren Sie einen Stackautomaten, der die durch die Grammatik

$$\begin{aligned} S &\to \varepsilon \\ &\to SS \\ &\to GSU \\ G &\to \mathtt{0} \mid \mathtt{2} \mid \mathtt{4} \mid \mathtt{6} \mid \mathtt{8} \\ U &\to \mathtt{1} \mid \mathtt{3} \mid \mathtt{5} \mid \mathtt{7} \mid \mathtt{9} \end{aligned}$$

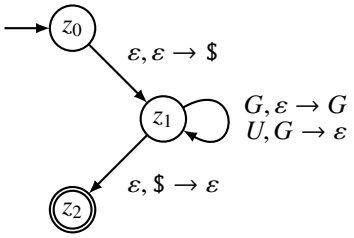
definierte Sprache akzeptiert.

Hinweis. Versuchen sie zuerst die Sprache zu verstehen, die von dieser Grammatik erzeugt wird und erstellen Sie dann einen "passenden" Stackautomaten.

Lösung. Die Wörter der Sprache entstehen aus korrekt geschachtelten Klammer-Ausdrücken, indem man die öffenenden Klammern durch gerade Ziffern, die schließenden Klammern durch ungerade Ziffern ersetzt. Da zwischen den öffnenden und schließenden Ziffern kein weiterer Zusammenhang besteht, genügt es auf dem Stack über die Anzahl der geraden und ungeraden Ziffern Buch zu führen.

Der Automat legt also ein Symbol G auf den Stack, wenn eine gerade Ziffer gelesen wird, und entfernt ein Symbol G vom Stack, wenn eine ungerade Ziffer gelesen wird. Ein Wort wird akzeptiert, wenn am Ende des Wortes der Stack leer ist.

Als Zustandsdiagramm kann man



verwenden.

Alternativ, jedoch einiges aufwendiger, kann auch der Algorithmus zur Konstruktion eines PDA aus einer Grammatik verwendet werden, der in Satz 7.4 skizziert worden ist. ○

7.3. Konstruieren Sie einen Stackautomaten, der die Sprache

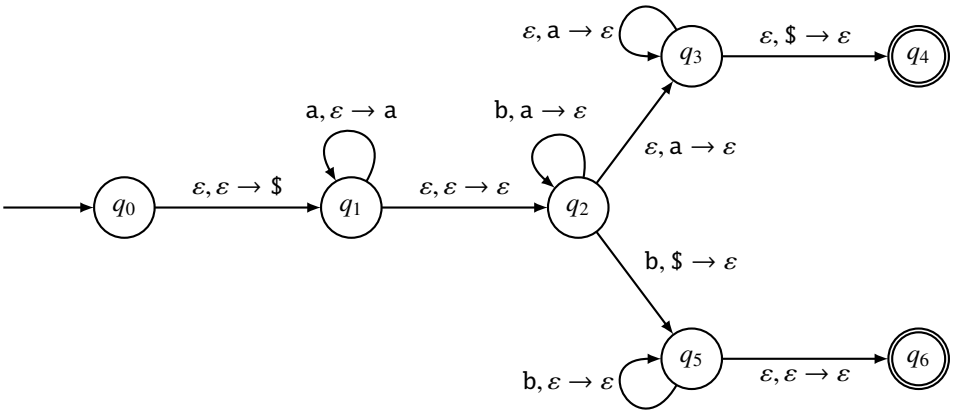
$$L = \{\mathtt{a}^i\mathtt{b}^j \mid i \neq j\}$$

akzeptiert.

Lösung. Der PDA funktioniert nach dem gleichen Prinzip wie der in Abschnitt 7.1.3 besprochene PDA, der die Sprache $\{\mathtt{0}^n\mathtt{1}^n | n \geq 0\}$ akzeptiert.

Wenn $i > j$ ist, dann ist es am Ende des Wortes noch möglich, mindestens ein weiteres a vom Stack zu lesen, bevor das Stackende-Zeichen gefunden wird.

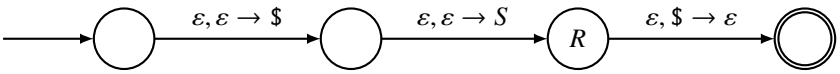
Wenn $i < j$ ist, dann kann nach dem erreichen des Stackende-Zeichens noch ein b vom Input gelesen werden, und weitere b können gelesen werden, ohne dass der Stack modifiziert wird.



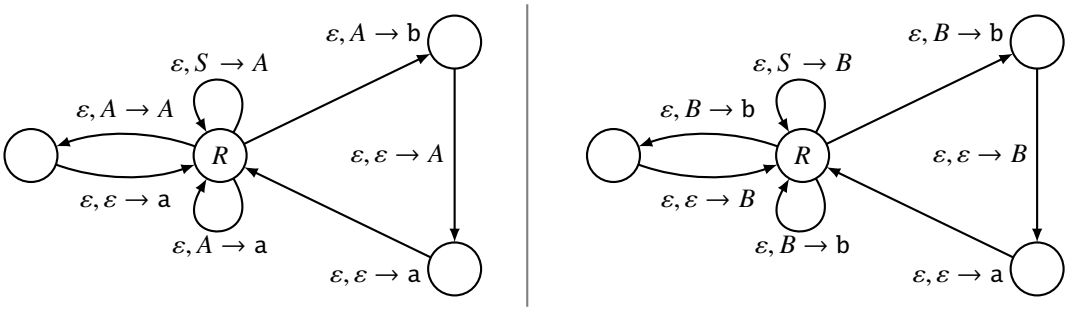
Alternativ könnte man den Stackautomaten dadurch finden, dass man zuerst eine Grammatik konstruiert und diese dann mithilfe von Satz 7.4 in einen Stackautomaten umwandelt. Die Grammatik muss Wörter mit einem Überschuss an a oder b produzieren, wir schreiben die Variable A für Wörter mit einem Überschuss an a und B für Wörter mit einem Überschuss an b. Wörter mit einem Überschuss an a entstehen, in dem man solche Wörter immer im a links wachsen lässt oder wenn man ein b rechts anfügt auch ein b links hinzufügt. Die Grammatik wird damit

$$\begin{aligned} S &\to A|B \\ A &\to \mathtt{a}A \\ &\to \mathtt{a}A\mathtt{b} \\ &\to \mathtt{a} \\ B &\to B\mathtt{b} \\ &\to \mathtt{a}B\mathtt{b} \\ &\to \mathtt{b} \end{aligned}$$

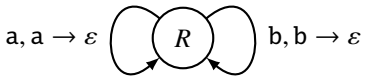
Die Konstruktion von Satz 7.4 liefert jetzt zunächst das Gerüst des Automaten



dem Zustand R fügt man jetzt entsprechend der Regeln folgende Übergänge hinzu. Für die Regeln mit A sind es die Übergänge im linken Zustandsdiagramm, für die Regeln mit B sind es die im rechten:



Zusätzlich braucht es noch die Regeln, die zur Verarbeitung der Eingabezeichen nötig sind:



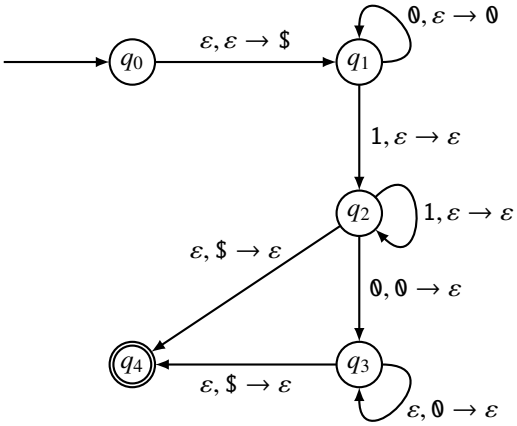
○

7.4. Finden Sie einen Stackautomaten, der

$$L = \{\mathtt{0}^i\mathtt{1}^j\mathtt{0}^i \mid j > 0 \wedge i \geq 0\}$$

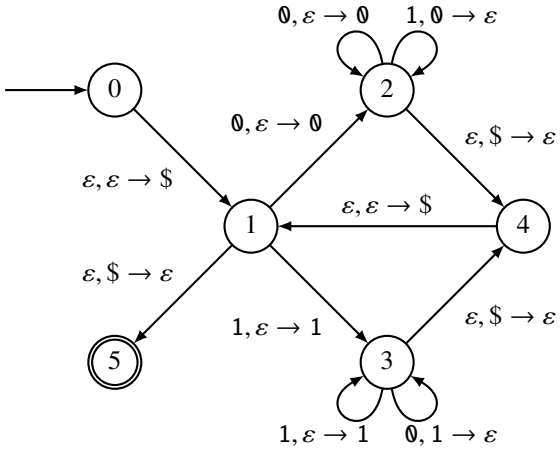
akzeptiert.

Lösung. Man kann zum Beispiel den folgenden fast[1] deterministischen Stackautomaten verwenden, um die Sprache zu akzeptieren:



○

7.5. Man finde eine Grammatik für die Sprache $L = \{w \in \Sigma^* \mid |w|_1 = |w|_{\mathtt{0}}\}$ über dem Alphabet $\Sigma = \{\mathtt{0}, 1\}$, indem man sie aus dem folgenden Stackautomaten ableitet:



Hinweis. Die Konstruktion dieses Automaten wird in der Lösung von Verständniskontrolle 7.1 erläutert. Die Idee des Automaten ist, dass der obere Zweig mit dem Zustand 2 verwendet wird, solange mehr $\mathtt{0}$ als 1 gelesen wurden. Der untere Zweig mit dem Zustand 3 wird dagegen verwendet, wenn mehr 1 als $\mathtt{0}$ gelesen wurden. Im oberen Zweig werden die Nullen auf den Stack gelegt und der Stack wird mit Einsen wieder abgebaut. Im unteren Zweig werden die Einsen auf den Stack gelegt und der Stack wird mit Nullen wieder abgebaut. Es ist also möglich, dass während der Verarbeitung

[1] Der Stackautomat wird deterministisch, wenn man fehlende Übergängen zu einem "Fehlerzustand" ergänzt.

eines Wortes mehrmals zwischen dem oberen und unteren Teilautomaten hin- und hergewechselt wird. Akzeptabel ist ein Wort, welches den Stack leer lässt. Im Zustand 1 und 4 sind immer gleich viele Einsen wie Nullen gelesen worden.

Lösung. Dieser Stackautomat erfüllt bereits alle Voraussetzungen, damit der Algorithmus aus dem Unterricht angewendet werden kann. Es gibt nur einen Akzeptierzustand und der Stack wird vor dem Akzeptieren eines Wortes geleert.

Der im Unterricht besprochene Algorithmus verwendet als Variablen der Grammatik die Symbole A_{pq}, die für Wörter stehen, die den Stackautomaten vom Zustand p in den Zustand q überführen, ohne den Stack zu verändern. Die Startvariable der so konstruierten Grammatik ist also A_{05}.

Zwischen 0 und 1 wird das Zeichen \$ auf den Stack gelegt, und es kann im Schritt zwischen 1 und 5 oder im Schritt zwischen 2 und 4 oder zwischen 3 und 4 wieder entfernt werden. Außerdem kann zwischen 4 und 1 wieder ein \$ auf den Stack gelegt werden. Daher heißen die ersten Regeln

$$\begin{aligned} A_{05} &\to \varepsilon A_{11} \varepsilon \\ &\to A_{04} A_{45} \end{aligned}$$

Ausgehend vom Zustand 1 gibt es gar keine Wege, die ohne den Stack zu verändern wieder bei 1 ankommen, denn jeder solche Weg müsste über Zustand 4 führen, wo ein Zeichen entfernt wird, welches nicht zwischen 1 und 4 auf den Stack gelegt wurde. Daher ist die einzige mögliche Regel für A_{11}

$$A_{11} \to \varepsilon.$$

Für die Variable A_{04} gibt es drei Regeln. Eine, die über den Zustand 2 führt, die zweite, die über den Zustand 3 führt, und schließlich eine, die die Möglichkeit von Zustand 4 nach 4 berücksichtigt:

$$\begin{aligned} A_{04} &\to \varepsilon A_{12} \varepsilon \\ &\to \varepsilon A_{13} \varepsilon \\ &\to A_{04} A_{44} \end{aligned} \tag{1}$$

Wegen des einzig möglichen Übergangs von 4 zu 1 ist die Menge der Wörter, die von 0 zu 4 führen können mit leerem Stack die gleiche ist wie die, die von 4 zu 4 führen können mit leerem Stack. Die erzeugten Wörter sind also Verkettungen von Wörtern, die aus A_{12} oder A_{13} erzeugt werden.

Für A_{12} und A_{13} wiederum bekommt man die Regeln

$$\begin{aligned} A_{12} &\to \mathtt{0} A_{22} \mathtt{1} \mid A_{12} A_{22} \\ A_{13} &\to \mathtt{1} A_{33} \mathtt{0} \mid A_{13} A_{33} \end{aligned} \tag{2}$$

Aus der Variablen A_{12} entstehen immer Wörter, die mit 0 beginnen, aus A_{13} entstehen immer Wörter, die mit 1 beginnen. Beim Parsen mit dieser Grammatik zeigt daher das nächste Zeichen immer an, in welche der Variablen A_{12} oder A_{13} man A_{04} auflösen muss.

Zusätzlich zu den ε-Regeln für die Variablen A_{22} und A_{33} findet man die Regeln, die ein Zeichen auf den Stack legen und am Schluss wieder abbauen:

$$\begin{aligned} A_{22} &\to \mathtt{0} A_{22} \mathtt{1} \mid A_{22} A_{22} \mid \varepsilon \\ A_{33} &\to \mathtt{1} A_{33} \mathtt{0} \mid A_{33} A_{33} \mid \varepsilon \end{aligned} \tag{3}$$

Die Regel für A_{22} erzeugt Wörter der Form $\mathtt{0}^n\mathtt{1}^n$, Verkettungen davon und sie schachtelt solche Wörter auch wieder zwischen 0 und 1 ein. Betrachtet man 0 als die öffnende "Klammer" und 1 als

die schließende "Klammer", dann steht A_{22} für einen korrekten Klammerausdruck. Dasselbe gilt für A_{33} mit vertauschten Rollen von $\mathtt{0}$ und $\mathtt{1}$.

Wir müssen noch die Variable A_{45} weiter entwickeln. Es gibt drei Wege, wie man von 4 nach 5 gelangen kann ohne den Stack zu verändern. Entweder auf dem direkten Weg oder mit einer Schleife über den Zustand 2 oder 3. Diesen Wegen entsprechen die Regeln

$$\begin{aligned} A_{45} &\to \varepsilon A_{11}\varepsilon \to \varepsilon \\ &\to A_{44}A_{45} \\ A_{44} &\to \varepsilon A_{12}\varepsilon \\ &\to \varepsilon A_{13}\varepsilon \\ &\to A_{44}A_{44} \\ &\to \varepsilon \end{aligned}$$

Diese Regeln drücken aus, dass A_{45} eine Verkettung von Wörtern ist, die jeweils aus A_{44} entwickelt werden können.

Damit ist die Grammatik gefunden, sie ist aber noch nicht sehr übersichtlich. Um sie zu vereinfachen, führen wir neue, lesbarere Variablen ein. Wir beginnen mit den Regeln (2) und (3), die wir als

$$\left.\begin{aligned} A_{12} &\to \mathtt{0}A_{22}\mathtt{1} \mid A_{12}A_{22} \\ A_{13} &\to \mathtt{1}A_{33}\mathtt{0} \mid A_{13}A_{33} \\ A_{22} &\to \mathtt{0}A_{22}\mathtt{1} \mid A_{22}A_{22} \mid \varepsilon \\ A_{33} &\to \mathtt{1}A_{33}\mathtt{0} \mid A_{33}A_{33} \mid \varepsilon \end{aligned}\right\} \quad\Rightarrow\quad \left\{\begin{aligned} A_{12} &\to \mathtt{0}N\mathtt{1} \mid A_{12}N \\ A_{13} &\to \mathtt{1}E\mathtt{0} \mid A_{13}N \\ N &\to \mathtt{0}N\mathtt{1} \mid NN \mid \varepsilon \\ E &\to \mathtt{1}E\mathtt{0} \mid EE \mid \varepsilon \end{aligned}\right.$$

schreiben können. Die ersten beiden Regeln auf der rechten Seite sagen, dass die Variablen A_{12} und A_{13} das leere Wort nicht erzeugen können. Dies ist gleichbedeutend damit, die ε-Regeln in letzten zwei Regeln wegzulassen, so dass

$$\begin{aligned} N &\to \mathtt{0}N\mathtt{1} \mid NN \mid \mathtt{01} \\ E &\to \mathtt{1}E\mathtt{0} \mid EE \mid \mathtt{10} \end{aligned} \tag{4}$$

übrig bleiben und wir $A_{12} = N$ und $A_{13} = E$ setzen können.

Die übrigen Regeln der Grammatik reduzieren sich jetzt auf

$$\left.\begin{aligned} A_{05} &\to \varepsilon \\ &\to A_{04}A_{45} \\ A_{04} &\to N \\ &\to E \\ &\to A_{04}A_{44} \\ A_{45} &\to \varepsilon \\ &\to A_{44}A_{45} \\ A_{44} &\to N \\ &\to E \\ &\to A_{44}A_{44} \\ &\to \varepsilon \end{aligned}\right\} \quad\Rightarrow\quad \left\{\begin{aligned} S &\to \varepsilon \\ &\to BC \\ B &\to N \\ &\to E \\ &\to BD \\ C &\to \varepsilon \\ &\to DC \\ D &\to N \\ &\to E \\ &\to DD \\ &\to \varepsilon \end{aligned}\right.$$

Die Regeln für C in der mittleren Grammatik sagen, dass C eine Folge von D ist, wobei auch das leere Wort zulässig ist. Zusammen mit den Regeln für D bedeutet das, dass C beliebige, möglicher-

weise leere Folgen von N oder E sind. Die Variable C erzeugt also die gleichen Wörter wie D, man kann also C durch D ersetzen und die Regeln für C weglassen.

$$\left.\begin{array}{ll} S \to \varepsilon \\ \quad \to BC \\ B \to N \\ \quad \to E \\ \quad \to BD \\ C \to \varepsilon \\ \quad \to DC \\ D \to N \\ \quad \to E \\ \quad \to DD \\ \quad \to \varepsilon \end{array}\right\} \quad \Rightarrow \quad \left\{\begin{array}{ll} S \to \varepsilon \\ \quad \to BD \\ B \to N \\ \quad \to E \\ \quad \to BD \\ \\ \\ \\ D \to N \\ \quad \to E \\ \quad \to DD \\ \quad \to \varepsilon \end{array}\right.$$

Die letzten vier Regeln erlauben, Folgen von N und E zu bilden, oder auch das leere Wort. Die drei Regeln für B sagen dasselbe, aber da weder N noch E das leere Wort produzieren können, kann auch B niemals das leere Wort erzeugen. Die Regel $S \to BD$ sagt dann, dass S einfach eine Folge von N oder E sein. Dies kann man einfacher durch die Regeln

$$\begin{array}{ll} S \to \varepsilon \\ \quad \to SF \\ F \to N \\ \quad \to E \end{array}$$

ausdrücken. Damit bleiben jetzt zusammen mit (4) die Regeln

$$\begin{array}{l} S \to SF \mid \varepsilon \\ F \to N \mid E \\ N \to NN \mid \mathtt{0}N\mathtt{1} \mid \mathtt{01} \\ E \to EE \mid \mathtt{1}E\mathtt{0} \mid \mathtt{10} \end{array}$$

als endgültige Form der Grammatik. ○