Kellerautomaten und Klammersprachen

[1)]

(A1) Vorüberlegungen

(A) Konstruiere einen endlichen Automaten, der die Sprache $L_{Klammer2}$ aller Klammerausdrücke der Tiefe 2 erkennt. Zur Sprache gehören z.B.

```
(()), ()(), (), (()())
```

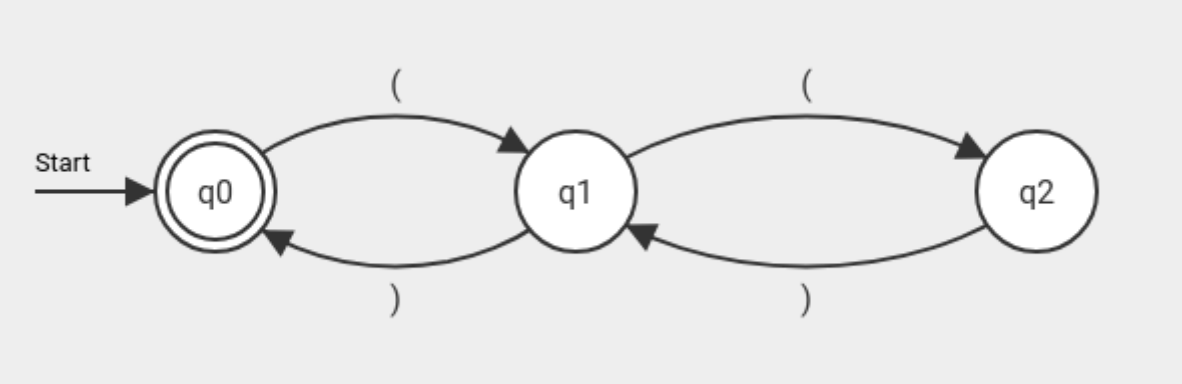
Nicht zur Sprache gehören z.B.

```
((()), ((())), ()), )(, )()(
```

(B) Gibt es einen endlichen Automaten A, der die Sprache $L_{ab} = \{(^n)^n \mid n = 1, 2, 3, ...\}$ erkennt? Wie sehen diese aus und worin unterscheiden sich die Automaten, die unterschiedliche Klammerungstiefen erkennen?

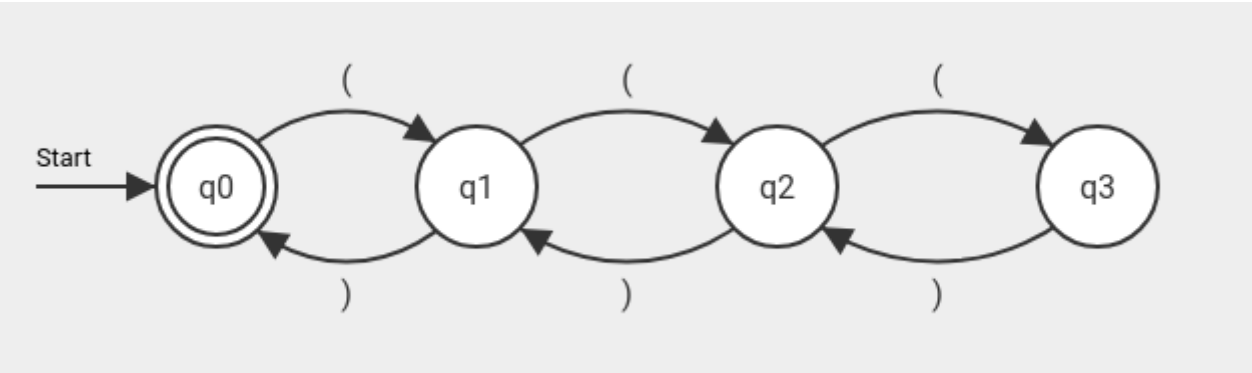
(C) Gib einen endlichen Automaten A an, der die allgemeine Klammersprache, also Klammerungen beliebiger Tiefe erkennt - siehst du ein Problem?

Lösung (A)



Lösung (B)

Ein Automat, der Klammerungen bis zur Tiefe 3 erkennen kann sähe so aus:



Für jede weitere Klammertiefe, die erkannt werden soll, muss ein weiterer Zustand hinzugefügt werden:

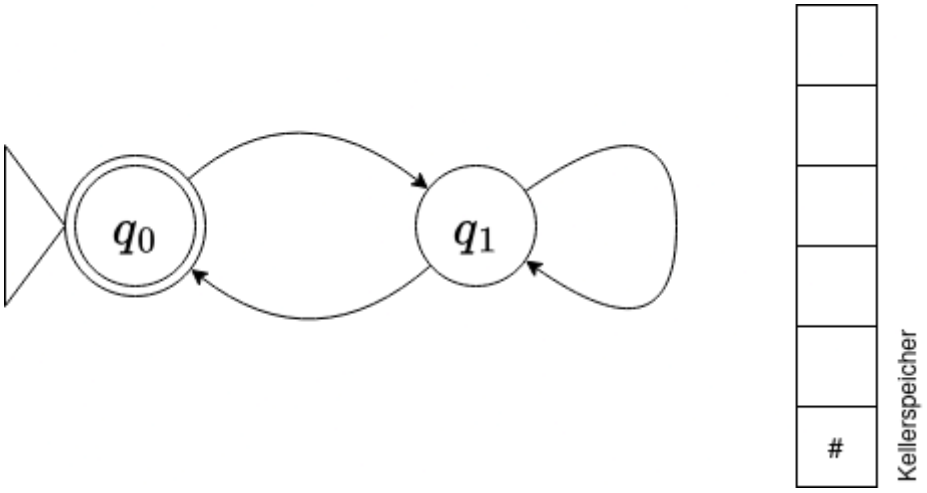
- Tiefe 2: https://flaci.com/A541hkqyky
- Tiefe 3: https://flaci.com/Afy17ed34i
- Tiefe 4: https://flaci.com/A5a9vz8j89

> Erkenntnis zu (C)
>
> Versuche, einen solchen endlichen Automaten A zu konstruieren, scheitern an der Unmöglichkiet, die Anzahl der öffnenden Klammern im Automaten mitzuzählen.
>
> Die **Endlichkeit** eines endlichen Deterministische Automaten besteht in der **Endlichkeit der Zustände** – um "zählen" zu können, wie viele Klammern bereits geöffnet wurden und damit auch wieder geschlossen werden müssen, braucht man aber genau einen Zustand mehr als die Klammertiefe beträgt. Ist diese aber beliebig, so ist kein Automat mit endlich vielen Zuständen konstruierbar.
>
> Wichtig ist der **Unterschied zur Eingabelänge** der gültigen Wörtern eine Sprache, die durch einen endlöichen Automaten erkannt werden kann: Diese kann sehr wohl unendlich lang sein – das äußert sich lediglich dartin, dass der Automat in Schliefen wiederholt duchlaufen wird, bis er am Ende des Eingabeworts an einen akzeptierenden Endzustand gelangt.

Der Kellerautomat am Beispiel

Wir erweitern den endlichen Automaten um einen **Speicher**, der (potentiell) unbegrenzt viele Daten aufnehmen kann. Für kontextfreie Sprachen reicht als Speicher ein **Stack** – dieser wird auch als "Keller" bezeichnet, aus diesem Grund nennt man solche um einen Stack erweiterten DEAs "Kellerautomat".

Für unsere Klammersprache könnte ein erster Entwurf eines Kellerautomaten so aussehen - der Kellerspeicher ist hier mit 6 "Speicherplätzen" gezeichnet, kann aber durch weiteres "auf den Stack legen" beliebig erweitert werden:



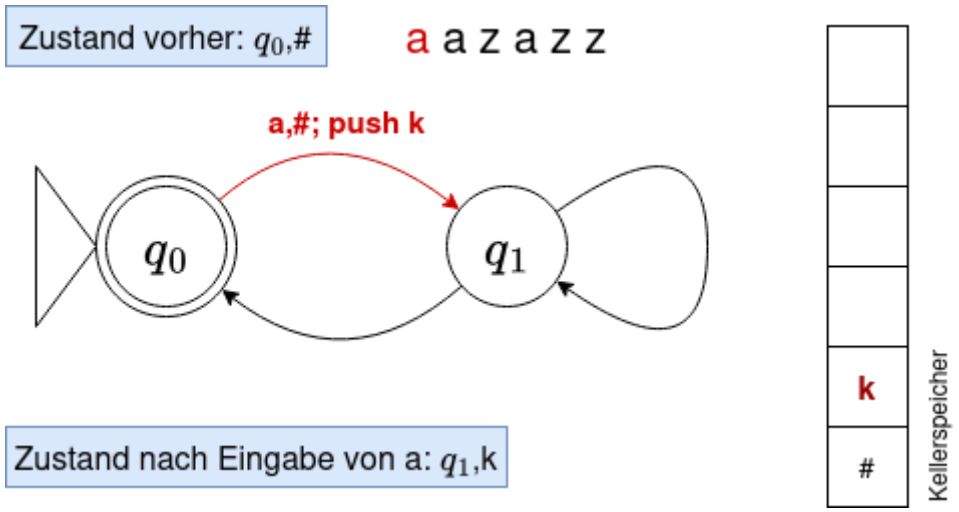
Zu Beginn ist der Kellerspeicher leer - das wird durch das Symbol **#** gekennzeichnet - der Ablauf beginn beim markierten Startknoten. Der Automat befindet sich also zu Beginn im Zustand **q_0,#**

Um Komplikationen beim Aufschreiben der Klammern zu vermeiden, verwenden wir von nun an für eine öffnende Klammer das Symbol **a**, für eine schließende Klammer ein **z**. Wir wollen nun nachvollziehen, ob das Wort **aazazz** vom Automaten akzeptiert wird - und vor allem, was der Automat dabei alles "tun" muss.

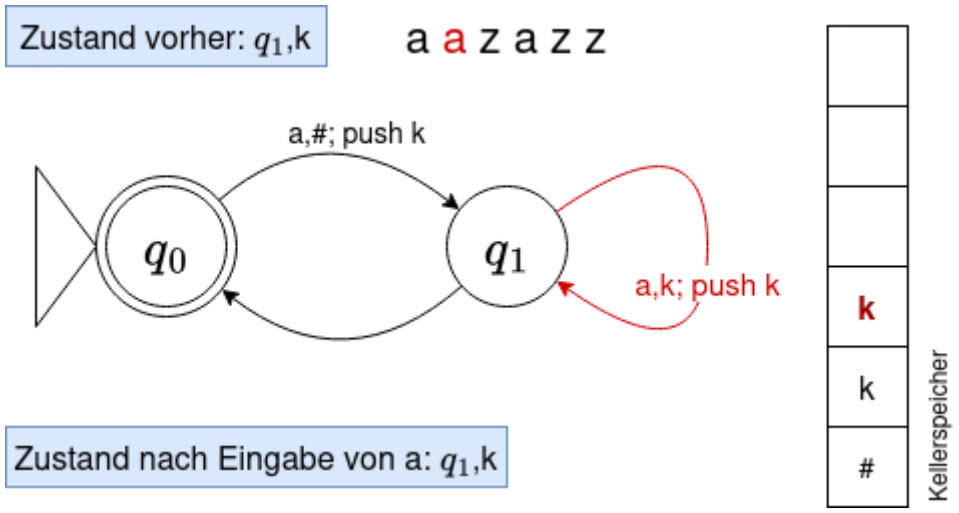
- Wir müssen bei jedem Übergang von einem Zustand in einen anderen nun auch noch unseren Speicher "bedienen". Der Speicher kennt 3 Operationen:
  - Etwas auf den Stack legen **push**
  - Das oberste Element vom Stack herunter nehmen **pop**
  - Den Stack unverändert belassen **noop**
- Welche Speicheroperation wir bei einem Zustandsübergang ausführen hängt nicht nur von der Eingabe ab, sondern kann außerdem vom Zustand des Stacks abhängen. Ist der Stack leer, welches Element liegt oben?

Schritt für Schritt - wir bauen einen Kellerautomaten

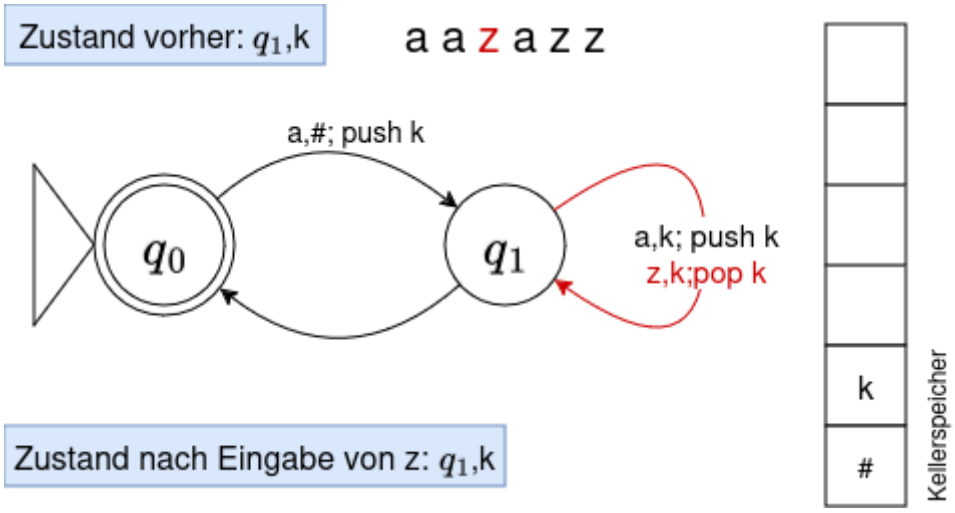
Mit einem z kann der Ausdruck nicht beginnen, das führt direkt zu einem Fehler - die Frage ist also, was geschehen muss, wenn bei leerem Stack in Zustand q_0 ein **a** eingegeben wird - wie muss der obere Übergang von q_0 nach q_1 beschriftet werden? Der Ausgangzustand ist **q_0,#**, der nächste Zustand ist q_1 - die Eingabe ist in diesem Fall also **a,#**, "Klammer auf, Stack leer". Welche Operation führen wir mit dem Speicher aus, um uns zu merken, dass wir eine Klammer geöffnet haben? → Wir legen eine Klammer **k**[2)] auf den Stack, um zu speichern, dass wir in der ersten Klammerebene sind: **push k**.



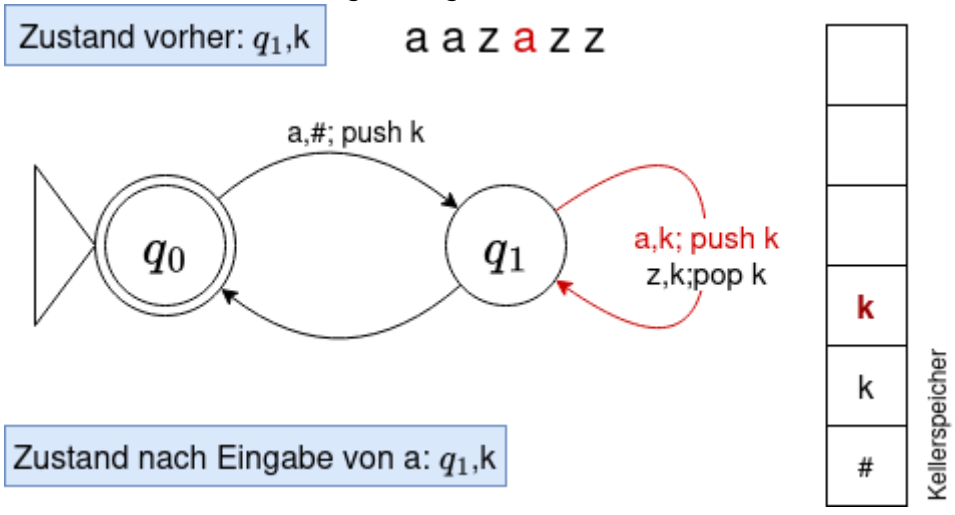
Wir befinden uns jetzt also in Zustand **q_1,k**. Wenn wir jetzt noch einmal eine öffnende Klammer eingeben, erhöhen wir die Klammertiefe, wir müssen also bei q_1 bleiben, allerdings legen wir ein weiteres k auf den Stack - wir merken uns damit, dass wir in der 2. Klammerebene sind.



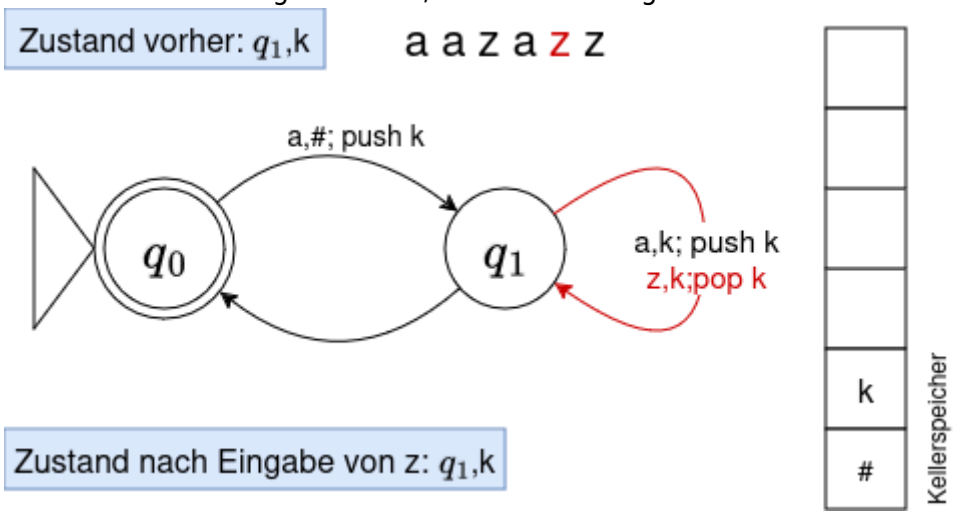
Wenn wir nun in Zustand q_1,k die erste schließende Klammer z eingeben gelangen wir nicht in einen gültigen Endzustand, also nicht nach q_0, sondern wieder nach q_1, müssen die Klammerungstiefe aber um eins reduzieren - das machen wir, indem wir eine Klammer k vom Stack entfernen.



Solange mindestens ein k auf dem Stack liegt, müssen wir jetzt keine neuen Übergänge und Kelleroperationen mehr festlegen: Jedes a legt ein k auf den Stack, jedes z entfernt eines - damit wird Buch über die Klammerungstiefe geführt.

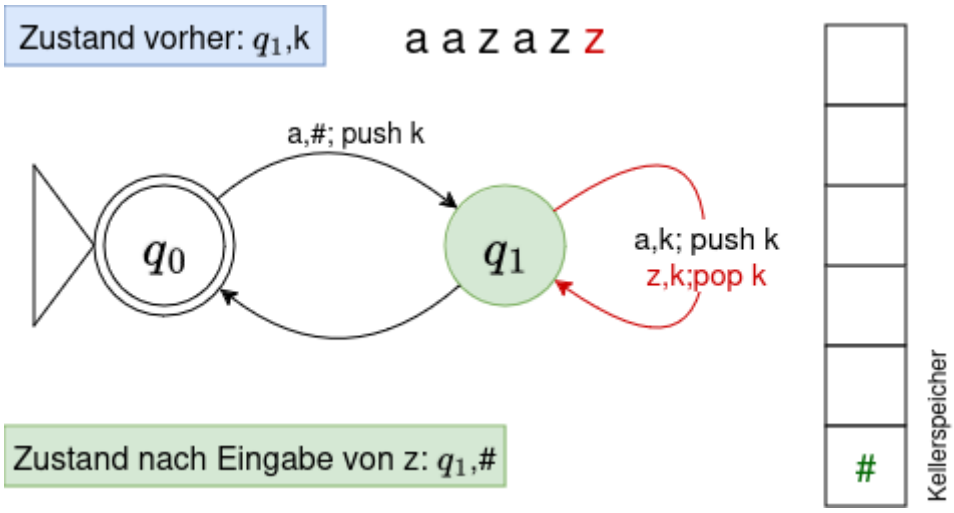


Es wird wieder ein k vom Stack genommen, die Klammerungstiefe reduziert.

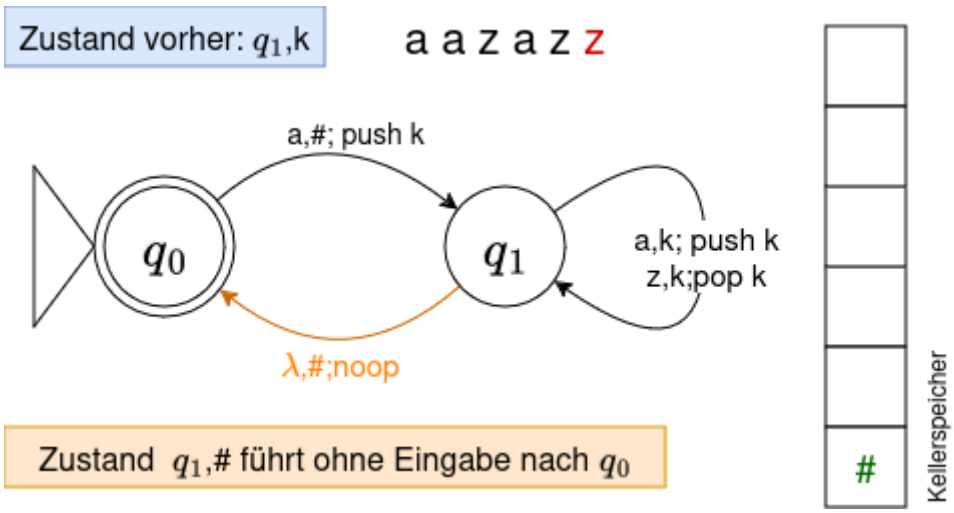


Mit der letzen schließenden Klammer wird das letzte k vom Stack entfernt, der Zustand des Automaten ist jetzt **q_1,#**.

Das stellt ein kleines Problem dar: Dieser Zustand signalisiert, dass das Eingabewort vom Automaten akzeptiert wird, denn hier kommen wir nur dann an, wenn wir gleich viele Klammern geöffnet haben wie wir geschlossen haben - **q_1,#** ist gewissermaßen ein akzeptierender Endzustand, q_1 alleine kann aber keiner sein.



Um diese Situation aufzulösen und berücksichtigen zu können, dass q_1 bei leerem Stack ein Endzustand ist, kann man einen Übergang definieren, der kein Eingabezeichen verbraucht, also bei λ "automatisch" ausgeführt wird, wenn der Stack einen bestimmten Zustand hat. Es dürfen dabei aber keine mehrdeutigen Übergänge entsteheen, da der Automat andernfalls nicht mehr deterministisch wäre.



Formale Darstellung eines Kellerautomaten

Ein Kellerautomat ist ein 6 Tupel, besteht also aus 6 Bestandteilen A={∑,Q,s,E,δ,Γ}[3)]

Allgemein	**Am Beispiel**
Eingabealphabet ∑	∑ = {a, z}
Menge Q von Zuständen	Q = {q0,q1,qF} (incl. Fehlerzustand!)
Startzustand s ∈ Q	s = q0
E ⊆ Z von Endzuständen	E = {q0}
Übergangsfunktion δ hängt meist von mehreren Krtiterien ab	Hängt von drei Kriterien ab, muss deswegen zeilenweise angegeben werden (s.u.)
Stackalphabet Γ (Gamma)	Γ ={#, k}; das # steht für einen leeren Stack

Die Übergangsfunktion δ für unser Beispiel kann zeilenweise wie folgt angegeben werden:

Kriterien				**Auswirkung**	
alter Zustand?	Eingabe?	auf Stack?	→	neuer Zustand	Operation
q_0	a	#	→	q_1	push k
q_1	a	k	→	q_1	push k
q_1	z	k	→	q_1	pop k
q_1	λ	#	→	q_0	nop

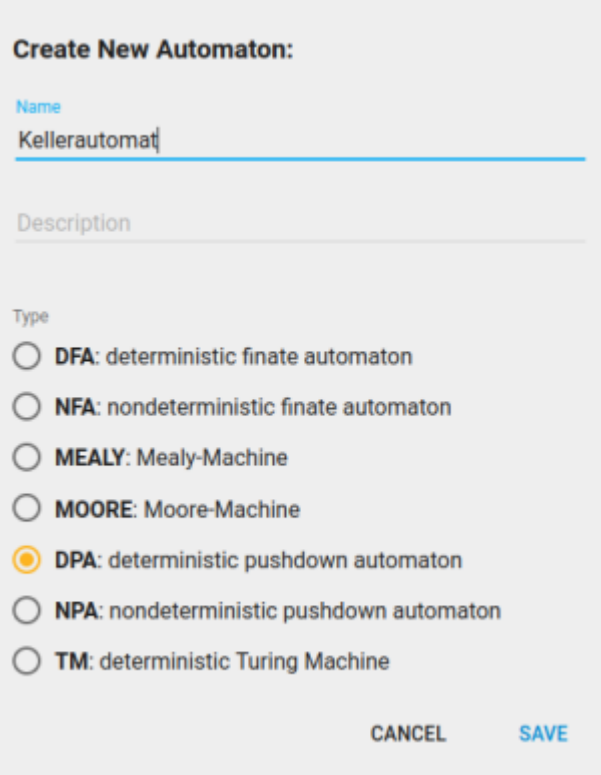
Ablaufprotokoll eines Kellerautomaten

Der "Lauf" eines Kellerautomaten bei der Verarbeitung einer Eingabe kann in einem Ablaufprotokoll dokumentiert werden, wobei die durchlaufenen Zustände, die Entwicklung des Stack und das Verbrauchen der Eingabe sichtbar werden sollten. Dazu eignet sich beispielsweise die Darstellung als Tabelle - hier für unsere Beispieleingabe **aazazz**:

Stack			k		k			
		k	k	k	k	k		
	#	#	#	#	#	#	#	#
Zustand	q0	q1	q1	q1	q1	q1	q1	q0
Vom Übergang verbrauchtes Eingabezeichen		a	a	z	a	z	z	

Eingabe endet, q0 ist Endzustand: Wort wird akzeptiert!

(A2)



- Simuliere den Klammerautomaten in FLACI.
- Teste den Automaten mit verschiedenen Eingaben und vollziehe seine Funktionsweise nach.

Hinweise

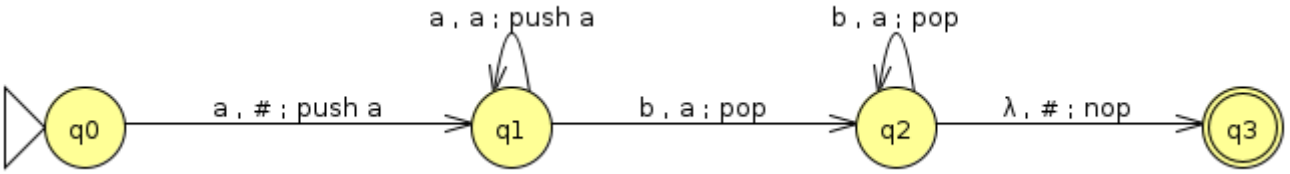
Flaci verwendet eine geringfügig andere Darstellung als unser Beispiel, außerdem stellt es die Stack Operationen nicht direkt zur Verfügung.

- Du musst einen DPA oder einen DKA erzeugen, je nach Spracheinstellung deines Browsers.
- Bei der Darstellung des Übergangs verwendet Flaci folgende Konvention:
  `(Stacksymbol, Eingabezeichen):Erstetzung für das Stacksymbol`
  - `(X,g):XX`: **push X**. Wenn das oberste Element des Stacks ein X ist und ein g eingegeben wird, wird X durch XX ersetzt, es wird also ein X auf den Stack gelegt.
  - `(X,b):ε`: **pop**. Wenn das oberste Element des Stacks ein X ist und ein b eingegeben wird, wird X durch "nichts" (ε) ersetzt.
  - `(X,p):X`: **noop**. Wenn das oberste Element des Stacks ein X ist und ein p eingegeben wird, wird X durch X ersetzt, es passiert also nichts.

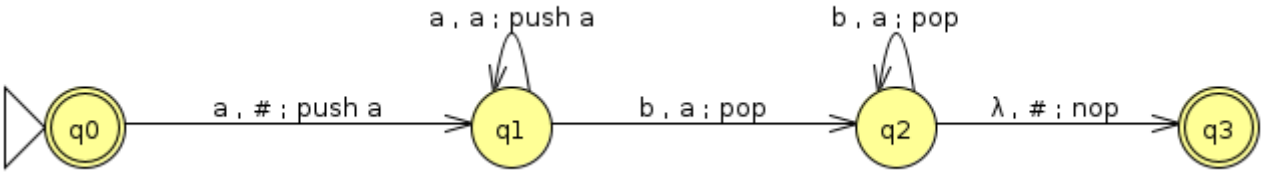
(A3)

Gegeben sind folgende Kellerautomaten. Beschreibe für jeden der Automaten die Wörter, die dieser erkennt und erstelle jeweils Ablaufprotokolle für zwei aussagekräftige Beispieleingaben – ein Wort, das akzeptiert wird und eines, das verworfen wird.

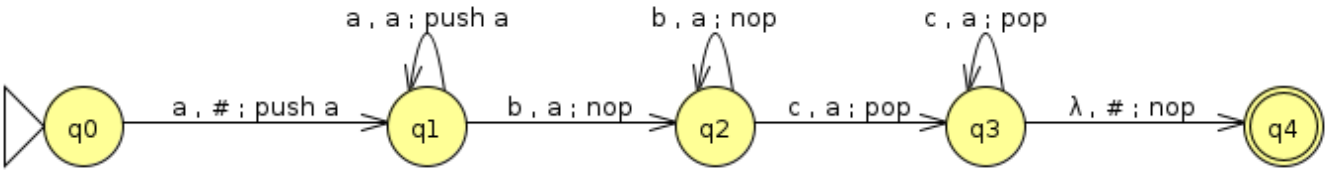
(A)



(B)



(C)



Lösung (A)

L = { $a^n b^n$ | n > 0 } / Wörter mit einer Anzahl n von "a" gefolgt von derselben Anzahl "b".

Lösung (B)

L = { $a^n b^n$ | n ≥ 0 } Wörter mit einer Anzahl "a" gefolgt von derselben Anzahl "b" – das leere Wort ist Teil der Sprache

Lösung (C)

(A4)

Entscheide mit Hilfe eines Kellerautomaten, ob die folgenden Sprachen kontextfrei sind. Wenn die Sprache kontextfrei ist, dann erläutere, wie der Kellerautomat arbeiten würde. Falls nicht, dann begründe, warum sie nicht mit einem Kellerautomaten erkannt werden kann.

- **(A)** L = { $a^n b^n$ | n ≥ 0 }
- **(B)** L = { $a^n b^n c^n$ | n ≥ 0 }
- **(C)** L = { w ∈ {a, b}* | #a = #b } die alle Wörter enthält, die aus gleich vielen a und b bestehen, die – im Gegensatz zur Sprache aus Aufgabenteil a) - in beliebiger Reihenfolge auftreten dürfen

Material

- Präsentation: Kellerautomaten (PDF)[4] (Code)

[1]
Diese Seite basiert auf der Arbeit von Dietrich/Lautebach und steht unter einer CC BY-NC-SA Lizenz
[2]
Warum reicht es ein k auf den Stack zu legen und nicht a und z?
[3]
Gelegentlich, z.B bei Flaci, wird das Zeichen für den leeren Stack als 7. Element des Tupels extra angegeben
[4]
Link öffnen, dann aus dem Browser in ein PDF drucken

From:
https://www.info-bw.de/ -

Permanent link:
https://www.info-bw.de/faecher:informatik:oberstufe:automaten:kellerautomaten:start?rev=1732025087

Last update: **19.11.2024 14:04**