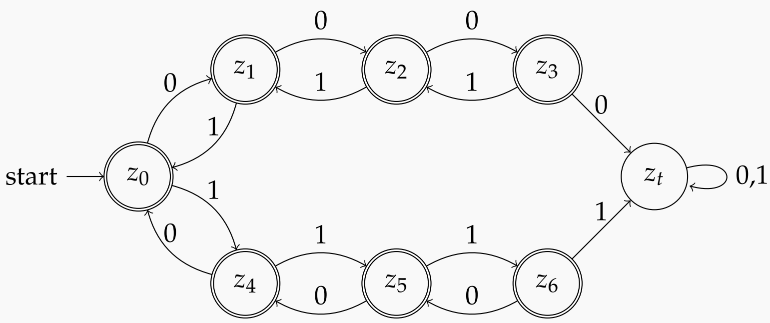
Aufgabe 1

Die Sprache L über den Alphabet $\Sigma = \{0, 1\}$ enthält alle Wörter, bei denen beim Lesen von links nach rechts der Unterschied in der Zahl der 0en und 1en stets höchstens 3 ist. Also ist $w \in L$ genau dann, wenn für alle u, v mit $w = uv$ gilt $||u|_0 - |u|_1| \leq 3$. Erinnerung: $|w|_a$ bezeichnet die Zahl der a's im Wort w.

(a) Sei $A = (Q, \Sigma, \delta, q_0, E)$ ein deterministischer endlicher Automat für L. Es sei $w_1 = 111$, $w_2 = 11$, $w_3 = 1$, $w_4 = \varepsilon$, $w_5 = 0$, $w_6 = 00$, $w_7 = 000$. Machen Sie sich klar, dass der Automat jedes dieser Wörter verarbeiten können muss. Folgern Sie, dass der Automat mindestens sieben Zustände haben muss. Schreiben Sie Ihr Argumentation schlüssig und vollständig auf.

> Ein deterministischer endlicher Automat hat keinen zusätzlichen Speicher zur Verfügung, in dem die Anzahl der bisher vorkommenden Einsen und Nullen gespeichert werden könnte. Ein deterministischer endlicher Automat kann die von der Sprache benötigten Anzahl an Einsen und Nullen nur in Form von Zustanden speichern. Um die Anzahl von 3 Einsen bzw. 3 Nullen zu speichern, sind also 6 Zustände nötig.
>
> Die Wörter 01 oder 0011 oder 0101 etc. haben eine Differenz von 0, wenn die Anzahl an Nullen und Einsen abzogen wird. Um auch diese Wörter darstellen zu können, ist mindestens ein weiterer Zustand nötig.

(b) Begründen Sie, dass L regulär ist.



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ait6va31c

(c) Jemand behauptet, diese Sprache sei nicht regulär und gibt folgenden „Beweis" dafür an: Wäre L regulär, so sei n eine entsprechende Pumping-Zahl. Nun ist $w = (01)^n \in L$. Zerlegt man nun $w = uxv$, wobei $u = 0$, $x = 1$, $v = (01)^{n-1}$, so ist zum Beispiel $ux^5v \notin L$, denn es ist $ux^5v = 01111101010101...$. Legen Sie genau dar, an welcher Stelle dieser „Beweis" fehlerhaft ist.

Exkurs: Pumping-Lemma für Reguläre Sprachen

Es sei L eine reguläre Sprache. Dann gibt es eine Zahl j, sodass für alle Wörter $\omega \in L$ mit $|\omega| \geq j$ (jedes Wort ω in L mit Mindestlänge j) jeweils eine Zerlegung $\omega = uvw$ existiert, sodass die folgenden Eigenschaften erfüllt sind:

(i) $|v| \geq 1$ (Das Wort v ist nicht leer.)

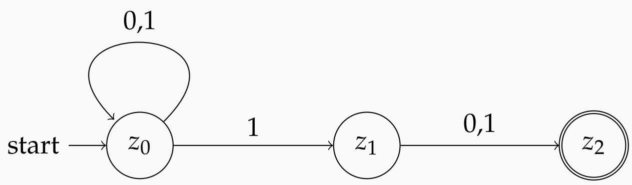
(ii) $|uv| \leq j$ (Die beiden Wörter u und v haben zusammen höchstens die Länge j.)

(iii) Für alle $i = 0, 1, 2, \ldots$ gilt $uv^iw \in L$ (Für jede natürliche Zahl (mit 0) i ist das Wort uv^iw in der Sprache L)

Die kleinste Zahl j, die diese Eigenschaften erfüllt, wird Pumping-Zahl der Sprache L genannt.

Das Wort $(01)^n$ wurde falsch zerlegt. Für die Pumping-Zahl $n = 3$ gibt es sehr wohl eine Zerlegung, die beim Aufpumpen regulär ist, also: $\omega = 010101$ ($u = 01, x = 01$ und $v = 01$). $ux^5v = 01010101010101 \in L$. Es gibt also eine Zerlegung, die beim Aufpumpen die 3 Pumping-Lemma-Eigenschaften erfüllt. Daher kann man das Pumping-Lemma so nicht widerlegt werden, indem man ein einziges Gegenbeispiel gibt.

(d) In anderen Fällen können nichtdeterministische endliche Automaten echt kleiner sein als die besten deterministischen Automaten. Ein Beispiel ist die Sprache $L_2 = \Sigma^*1\Sigma$ aller Wörter, deren vorletztes Symbol 1 ist. Geben Sie einen nicht-deterministischen Automaten mit nur drei Zuständen an, L_2 erkennt.

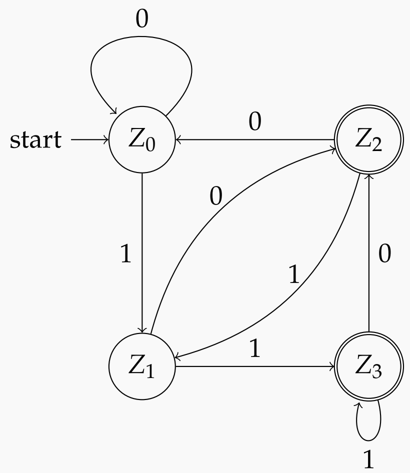


Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Apwzjufbg

(e) Führen Sie auf Ihrem Automaten die Potenzmengenkonstruktion und anschließend den Minimierungsalgorithmus durch. Wie viele Zustände muss ein deterministischer Automat für L_2 also mindestens haben?

Potenzmengenkonstruktion

Name	Zustandsmenge	Eingabe 0	Eingabe 1
Z_0	Z_0 $\{z_0\}$	Z_0 $\{z_0\}$	Z_1 $\{z_0, z_1\}$
Z_1	Z_1 $\{z_0, z_1\}$	Z_2 $\{z_0, z_2\}$	Z_3 $\{z_0, z_1, z_2\}$
Z_2	Z_2 $\{z_0, z_2\}$	Z_0 $\{z_0\}$	Z_1 $\{z_0, z_1\}$
Z_3	Z_3 $\{z_0, z_1, z_2\}$	Z_2 $\{z_0, z_2\}$	Z_3 $\{z_0, z_1, z_2\}$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ajfcofpb9

Minimierungsalgorithmus

Z_0	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
Z_1	x_2	$\emptyset$	$\emptyset$	$\emptyset$
Z_2	x_1	x_1	$\emptyset$	$\emptyset$
Z_3	x_1	x_1	x_2	$\emptyset$
	Z_0	Z_1	Z_2	Z_3

x_1 Paar aus End-/ Nicht-Endzustand kann nicht äquivalent sein.

x_2 Test, ob man mit der Eingabe zu einem bereits markiertem Paar kommt.

x_3 In weiteren Iterationen markierte Zustände.

x_4 ...

Übergangstabelle

Zustandspaar	**0**	**1**
(Z_0, Z_1)	(Z_0, Z_2) x_2	(Z_1, Z_3) x_2
(Z_2, Z_3)	(Z_0, Z_1)	(Z_1, Z_3) x_2

Wie aus der oben stehenden Tabelle abzulesen ist, gibt es keine äquivalenten Zustände. Der Automat kann nicht minimiert werden. Er ist bereits minimal.