Fragen, Antworten, Kommentare und Hinweise

Da es u. a. um die Verknüpfung von Online-Lehre mit klassischer Präsenzlehre geht, bitte dringend an der leicht erweiterten Lehrevaluation teilnehmen: https://evaluation.hs-osnabrueck.de/unizensus/de/sl/df4oT7KjDs13

Das Video zur Lösung der Aufgabe 39 finden Sie unter: https://youtu.be/lICMrmjcJHw . Natürlich dürfen in Ihrer Lösung zu 39 a) auch weitere Zustände vorkommen, auch wenn sie nicht nutzbar sind. Der (eigentlich auch unnötige) Zustand z2 ist Endzustand. Bei b fehlt bei der leeren Menge eine Schleife mit a, b.

Das Video zur Lösung der Aufgabe 40 finden Sie unter: https://youtu.be/zgL1RYheaD4 . In Aufgabe 40a) fehlt ein Übergang z1 a z0.

Frage: Müssen wir in der Klausur exakt nach den vorgegebenen Algorithmen vorgehen?

Antwort: Nur wenn dies im Detail, z. B, durch die Vorgabe einer zu füllenden Tabellenstruktur vorgegeben ist (Aufgaben 4 und 8). Wenn Sie Fehler machen, könnte es aber schwieriger sein Teilpunkte zu vergeben, wenn mir der verkürzte Rechenweg nicht klar oder die Abkürzung schlicht falsch ist.

Frage: Was wird sich an den Klausuraufgaben inhaltlich ändern?

Antwort: Gehen Sie davon aus, dass die Aufgabenstellungen identisch sind, nur alle Beispieldaten geändert werden. Da dadurch die Aufgaben etwas einfacher oder schwerer werden, können sich die Punktzahlen leicht ändern. Bei Fragen wie „gilt Eigenschaft XY“ kann es natürlich sein, dass diese Eigenschaft in der Probeklausur gilt, in der echten Klausur nicht oder umgekehrt.

Frage: Was passiert genau im Automat A3 in Folie 240 mit dem Wort a?

Antwort: Nach a kann man in z1 sein, so dass als nächste Zeichen b oder c möglich sind. Nach a kann man aber auch in z2 sein, da aε = a. Es wurde also intern nichtdeterministisch entschieden nach z2 zu gehen, danach kann als Zeichen nur b folgen. Ähnlich zu z2 kann man nach a auch in z3 sein und da kann nur c folgen.

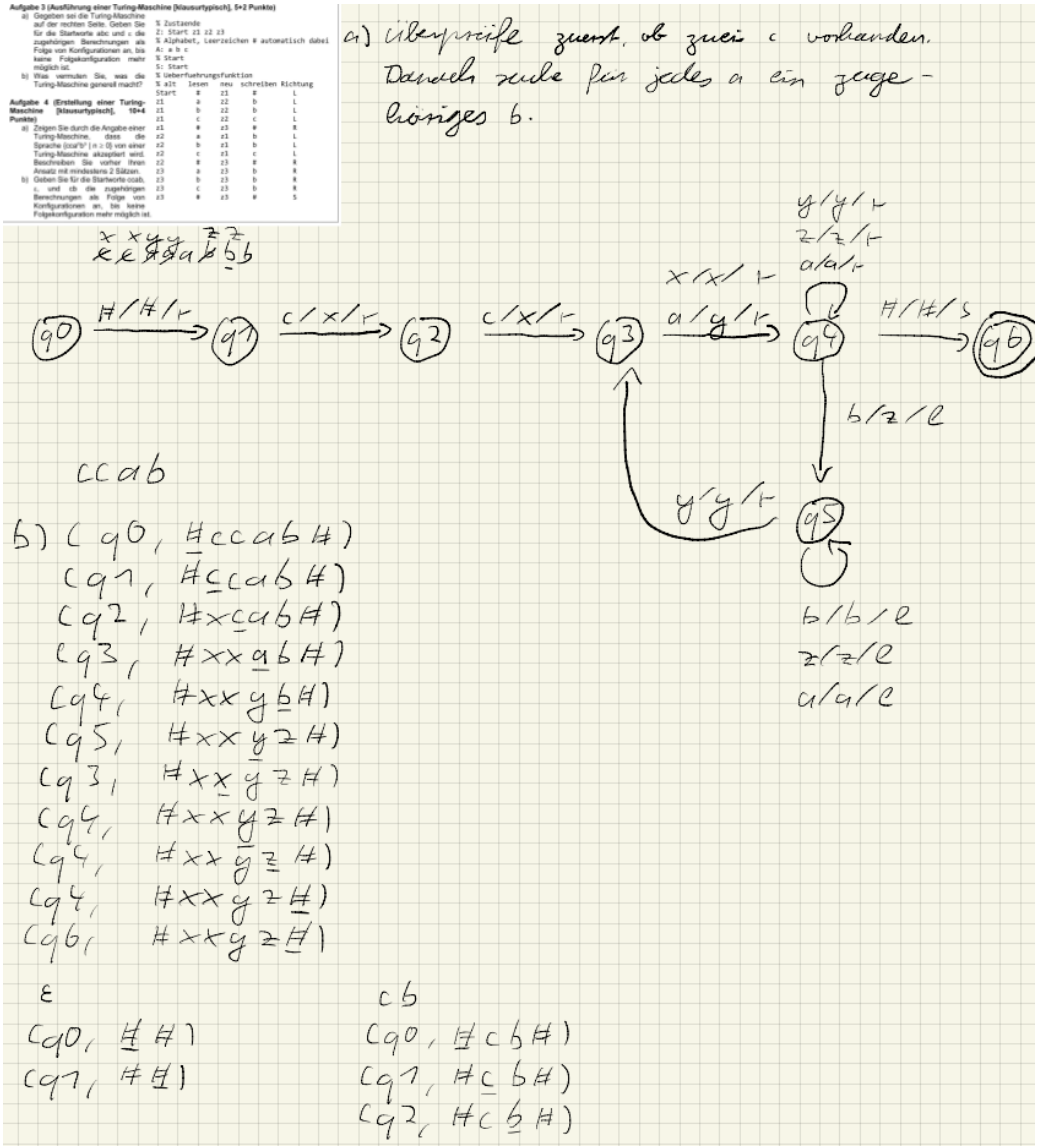
Frage: Ich verstehe nicht genau, was in den Tests zur Automaten-Erstellung passiert.

Antwort: Das ist wahrscheinlich, da hier, anders als sonst üblich nicht nur einzelne Beispiele, sondern die gesamte Struktur getestet, genauer verifiziert, wird. Bei einer Verifikation ist die Korrektheit nachgewiesen und nicht nur anhand von Beispielen plausibel gemacht worden. Um, nicht im Test die Lösung direkt einzubauen, wird ein anderer Ansatz zur Sprachbeschreibung benutzt. Dies sind

reguläre Ausdrücke, aus denen ein eindeutiger minimaler endlicher Automat berechnet wird, was allerdings erst etwas später in der Vorlesung vorkommt. Statt auf Lesbarkeit steht in diesem Fall die garantierte Korrektheit im Mittelpunkt.

Die Tests bei den Sprachen der Automaten haben damit den wesentlichen Vorteil, dass Sie die Lösungen vollständig auf Korrektheit überprüfen können. Das ist bei kontextfreien Grammatiken (und Turing-Maschinen) nicht möglich. Es gibt kein Programm was für gegebene beliebige zwei kontextfreie Grammatiken überprüfen kann ob sie die gleiche Sprache beschreiben. Dies ist als eines der unentscheidbaren Probleme bekannt. Das bedeutet auch, dass die gegebenen Tests nicht präzise sind. Wenn Sie eine Grammatik schreiben, die genau die Worte erzeugt, die in den Tests gefordert sind, wird diese Grammatik alle Tests erfüllen obwohl sie keinen tieferen Sinn hat.

Frage: Können Sie meine Lösung zu Aufgabe 4 überprüfen?

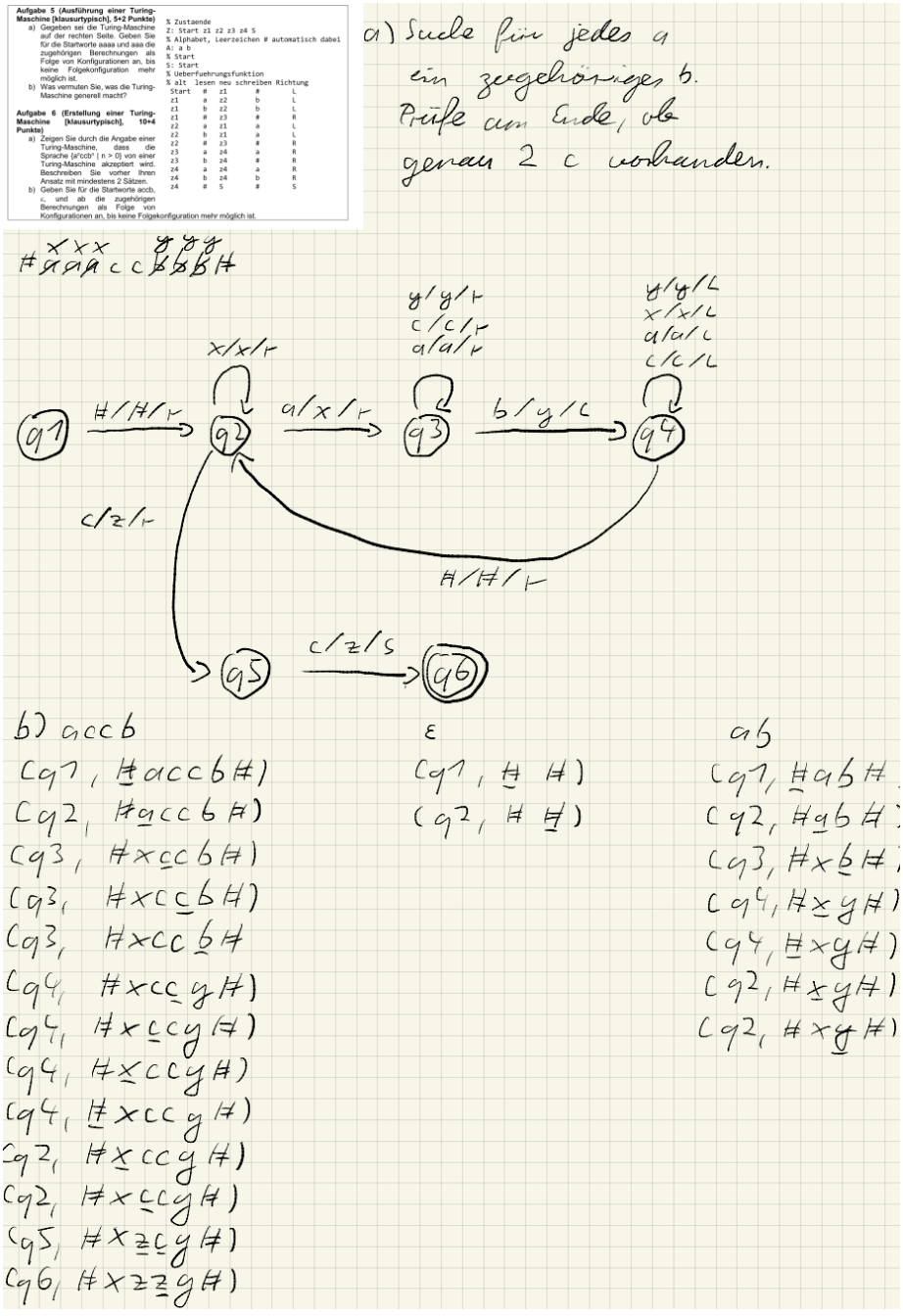


Antwort: Generell gerne, sollte aber wenn möglich eher in der Vorlesungs- oder der anschließenden Praktikumszeit geschehen, da man dann gemeinsam über die Ideen reden kann. In der VL-freien Zeit geht einfach eine E-Mail oder der Wunsch nach einem Zoom-Termin mit der Nennung möglicher Zeitintervalle. Meine Empfehlung ist immer, wenn möglich zuerst mit einem Simulator, entweder in meiner Umgebung oder einem im Netz auszuführen und vorhandene JUnit-Tests zu nutzen. Da der

Computer auch dann noch einfach „nein“ sagen kann, schicken Sie dann Ihren Lösungsansatz, eventuell mit Ergebnissen der genannten Tests an mich.

zu a) Die grobe Beschreibung wäre in Ordnung, wobei der zweite Satz recht unpräzise ist (was auch zu Problemen führt)

Zunächst fällt auf, dass sie davon ausgehen, dass die Turing-Maschine links vor dem Eingabewort startet. Das findet man häufig in der Literatur, ist aber in der Vorlesung (auch K. Morisse) nicht der Fall. Lässt sich leicht retten, indem man auf die linke der Seite der Eingabe läuft, was allerdings hier fehlt und Punkte kostet. Die Maschine ist fast ok, das Problem ist, dass sie nach einem a und der Suche nach einem b, wenn Sie sofort ein Leerzeichen finden (cca), terminieren. Das passiert auch bei cca^n, n>0 allgemein. Sie müssten in q5 prüfen, ob danach ein Leerzeichen kommt, um dann zu terminieren. Weiterhin muss das Wort cc akzeptiert werden, das ist bei Ihnen nicht der Fall. Der Rest ist ok.



Ihr Problem tritt auch in (q5,#xxyz#) auf, der von Ihnen genutzte Übergang existiert nicht im Diagramm. Die Idee, dass ein x auf das letzte gefundene a hindeutet, könnte zielführend werden.

Bei cb muss das c in der letzten Zeile ein x sein.

Generell würde Ihre Lösung in etwa die Hälfte der Punkte liefern.

Frage: Können Sie meine Lösung zu Aufgabe 6 überprüfen?

Die grobe Beschreibung wäre in Ordnung, wobei der zweite Satz recht unpräzise ist. (Startproblem s. vorherige Frage)

Die Maschine ist fast richtig, das Problem ist, dass zu viele b auch akzeptiert werden, da der Fehler nicht entdeckt wird. Sie akzeptieren a^nccb^m mit m≥n. Das Problem lässt sich beheben indem Sie prüfen, dass in q6 rechts davon kein b steht. Die Konfigurationsfolgen sind ok.

Generell würde Ihre Lösung in etwa 70% der Punkte liefern.

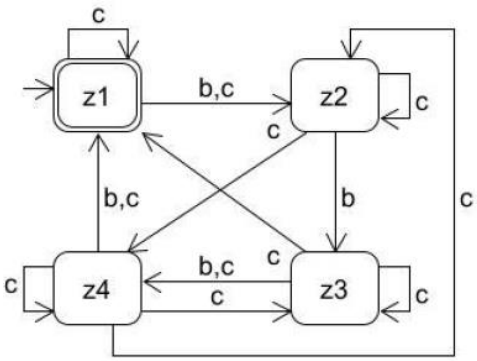
Allgemein ist es bei Strukturen der Form a^nb^n einfacher die äußeren a und b zu markieren und dann schrittweise nach innen zu gehen.

So ist meist feststellbar, ob die Anzahl gleich ist. Das wurde bei beiden Aufgaben nicht gemacht und hat zu Problemen geführt.

Frage: Wie funktioniert das mit dem deterministischen Automaten mit den 2^n Zuständen?

Antwort: Ist recht trickreich. Ein erster Teilansatz ist, dass es zunächst ein Zeichen gibt mit dem einfach von einem Zustand zum nächsten wieder zum Anfang geschaltet werden kann. Im Beispiel ist das das Zeichen b. Damit sind {z1}, {z2},{z3} und {z4} in der Potenzmengenkonstruktion erreichbar

Dann müssen durch die Potenzmengenkonstruktion alle weiteren Elemente der Potenzmenge erreicht werden können. Dazu nimmt man sich einen existierenden Zustand und ein neues Zeichen und geht mit diesem zu allen Zuständen der Potenzmenge. Im Beispiel



z1 -c-> {z1,z2} da mit b immer weitergeschaltet wird, ist damit auch {z2,z3}, {z3,z4}, {z4,z1} über die Potenzmengenkonstruktion erreichbar, da z. B. {z1,z2} -b-> {z2,z3}.

Es fehlt bei den zweielementigen Elementen der Potenzmenge noch {z1,z3} (und damit auch {z2,z4}). Dies könnte mit einem neuen Zeichen erreicht werden. Um die Zahl der Zeichen zu reduzieren reicht es, hier nicht z1 zu nutzen, also wird

z2 -c->{z2,z4} ergänzt.

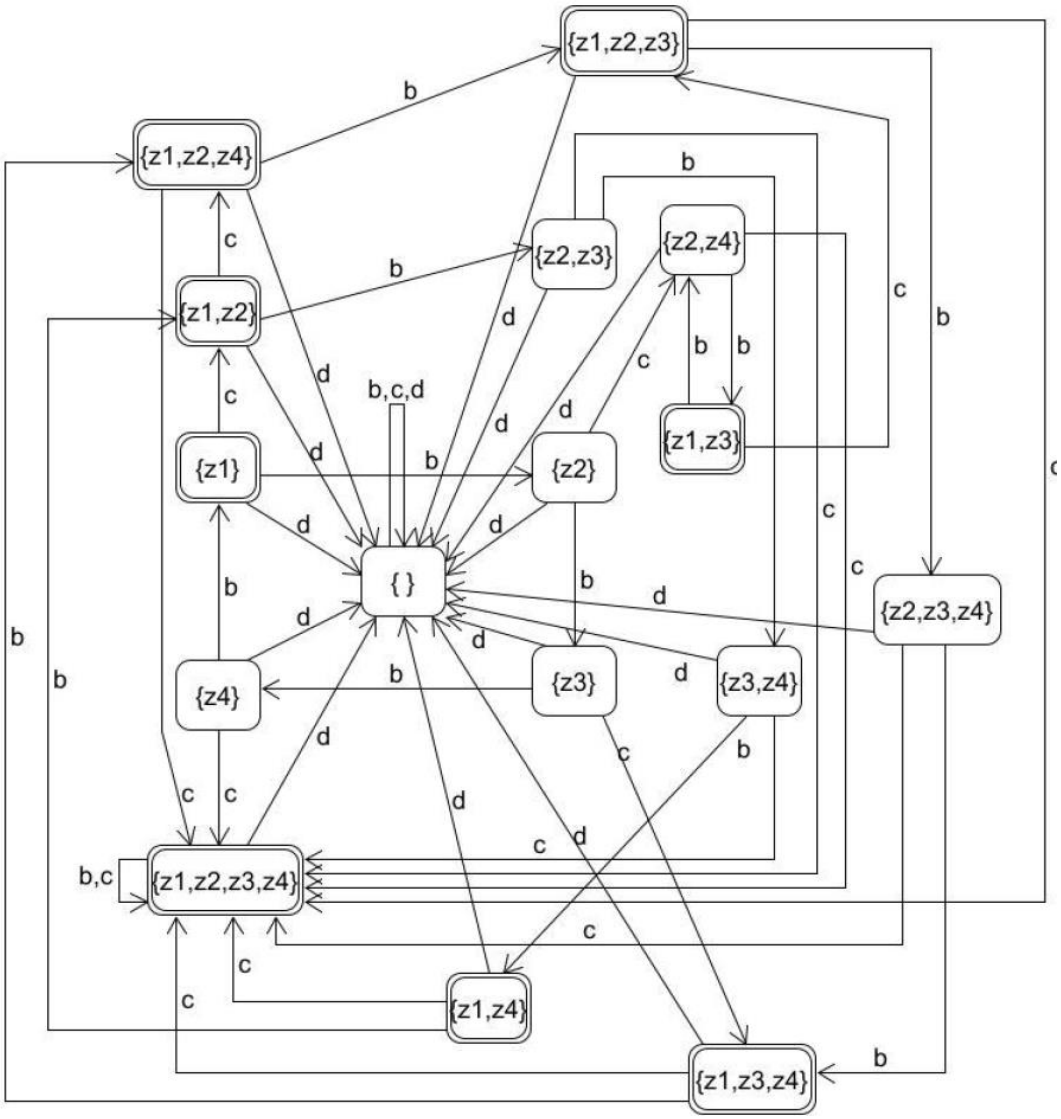
Nun werden alle dreielementigen Elemente der Potenzmenge betrachtet. Es wird noch kein neues Zeichen benötigt, da noch ein anderer Ausgangszustand nutzbar ist, es wird ergänzt:

z3 -c->{z1,z3,z4}, womit auch {z2,z4,z1}, {z3,z1,z2} und {z4,z2,z3} erreichbar sind.

Dann wird für das vierelementige Element eine Ergänzung vorgenommen, ein Zustand wurde noch nicht genutzt, so dass das gleiche Zeichen weiterhin nutzbar ist.

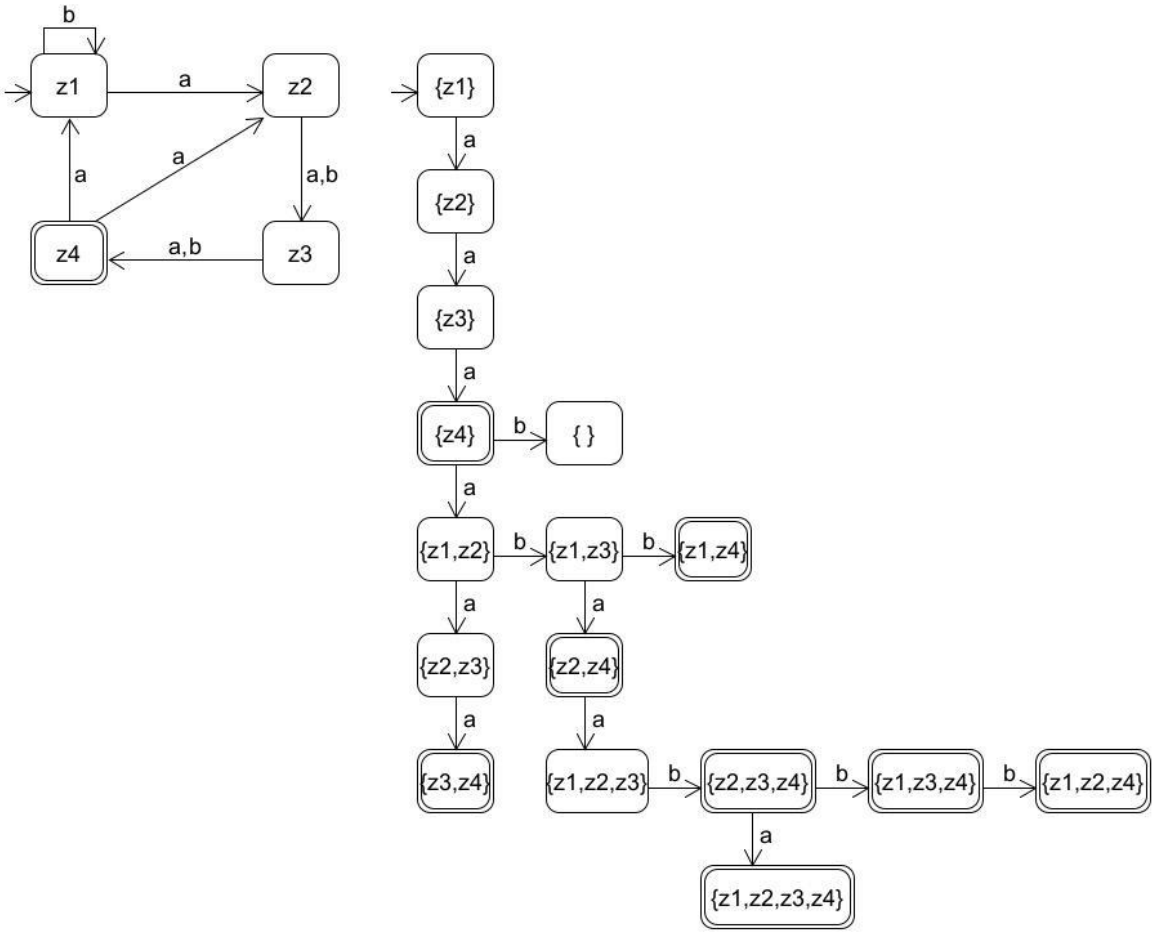
z4 -c->{z1,z2,z3,z4}

Es fehlt noch die leere Menge. Da alle Zustände für c schon ein eigenständiges Verhalten haben, wird ein neues Zeichen d benötigt. Dann muss es einen Zustand geben, der mit d in die leere Menge geht. Das wird hier vereinfachend für jeden Zustand gemacht, so dass d im Diagramm gar nicht vorkommt, aber zum Alphabet gehört. Der resultierende deterministische Automat sieht wie folgt aus.



Eine schönere Lösung findet sich bereits in dem folgenden Artikel, der innerhalb des Hochschulnetzes erreichbar ist.

F. R. Moore, On the Bounds for State-Set Size in the Proofs of Equivalence Between Deterministic, Nondeterministic, and Two-Way Finite Automata, IEEE Transactions on Computers, Volume C-20, Seiten 1211-1214, 1971, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1671701



Die vorherige Abbildung zeigt links den Automaten für 4 Zustände, weitere Zustände würden so wie die Zustände z2 und z3 ergänzt, die einfach mit a und b in den nachfolgenden Zustand übergehen. Der Anfangs- und der Endzustand würde sich wie beim gezeigten Automaten verhalten.

Auf der rechten Seite steht nicht der vollständige deterministische Automat, sondern nur typische Pfade, wie die Zustände erreicht werden, was die Konstruktionsidee verdeutlichen soll. Zunächst schaltet a nur einen Zustand weiter. Im Endzustand wird zu zwei Zuständen verzweigt, wodurch die Anzahl der Elemente der erreichten Menge von Zuständen sich um eins erhöhen kann. Um alle Mengen von Zuständen mit der gleichen Elementanzahl zu erreichen, hat z1 eine Schleife mit b, während alle anderen Nicht-Endzustände einen Zustand weiterschalten.

Frage: Bedeutet das Pumping-Lemma, dass wenn Worte aufpumpbar sind, die Sprache von einem Automaten akzeptierbar ist?

Antwort: Nein, die Formulierung fordert, dass dies für alle Worte ab einer bestimmten Länge gelten muss. Das Pumping-Lemma wird typischerweise nur eingesetzt, um zu zeigen, dass eine Sprache nicht von einem endlichen Automaten akzeptiert wird. Dazu muss dann nur ein einziges Wort gefunden werden, dass nicht aufpumpbar ist (da die Zustandsanzahl nicht bekannt ist, muss das Wort flexibel in der Länge angegeben werden). In VL 11 wird dies mit einem Beispiel mit der ersten Folie deutlich gemacht.

Im Beispiel geht es um die Sprache $\{a^n b^m \mid n < m\}$. Das Wort, das den Widerspruch zum Pumping-Lemma liefert ist $a^k b^{k+1}$. Es werden alle Zerlegungsmöglichkeiten w=xyz geprüft, genauer Formen von y auf Aufpumpbarkeit geprüft:

y kann aus mehreren a und mehreren b bestehen, durch einmal Aufpumpen wäre das Wort nicht mehr in der Sprache

y kann aus einem oder mehreren a bestehen, durch mehrfaches Aufpumpen wird n<m verletzt und das Wort liegt nicht in der Sprache

y kann aus einem oder mehreren b bestehen, da y auch weggelassen werden kann („schrumpfen“), wird n<m verletzt und das Wort liegt nicht in der Sprache.

Damit ist die Form eines nicht aufpumpbaren Wortes gefunden, damit erfüllt die Sprache nicht die Forderung des Pumping-Lemmas und damit kann die Sprache nicht von einem endlichen Automaten akzeptiert werden.