Formale Systeme, Automaten, Prozesse, SS 2020
Prof. Dr. P. Rossmanith
J. Dreier, T. Hartmann, H. Lotze, D. Mock

Datum: 4.5.2020

Übungsblatt 04

Aufgabe T12

Richtig oder falsch?

a) Es gibt eine reguläre Sprache, die von einem NFA mit 10 Zuständen akzeptiert wird, aber von keinem DFA mit 100 Zuständen.

b) Es gibt eine reguläre Sprache, die von einem NFA mit 10 Zuständen akzeptiert wird, aber von keinem DFA mit 1217 Zuständen.

c) Zwei minimale DFAs, welche jeweils komplementäre Sprachen akzeptieren, haben gleich viele Zustände.

d) Zwei minimale NFAs, welche jeweils komplementäre Sprachen akzeptieren, haben gleich viele Zustände.

Aufgabe T13

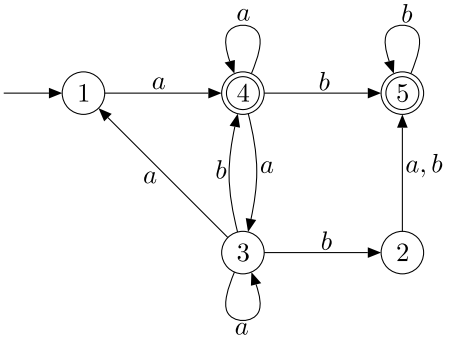
Zeigen Sie mit dem Satz von Myhill–Nerode, dass folgende Sprachen nicht regulär sind. Wenn es sich anbietet, können Sie auch mit Abschlußeigenschaften regulärer Sprachen arbeiten.

1. $L_1 = \{ a^i b^j c^k \mid i = j \text{ oder } j = k \}$
2. $L_2 = \{ a^i b^j \mid i = j \}$
3. L_3, die Menge korrekter Python-Programme
4. $L_4 = \{ u\$v\$w \mid u, v, w \in \{0,1\}^* \text{ und } \mathrm{bin}(u) + \mathrm{bin}(v) = \mathrm{bin}(w) \}$

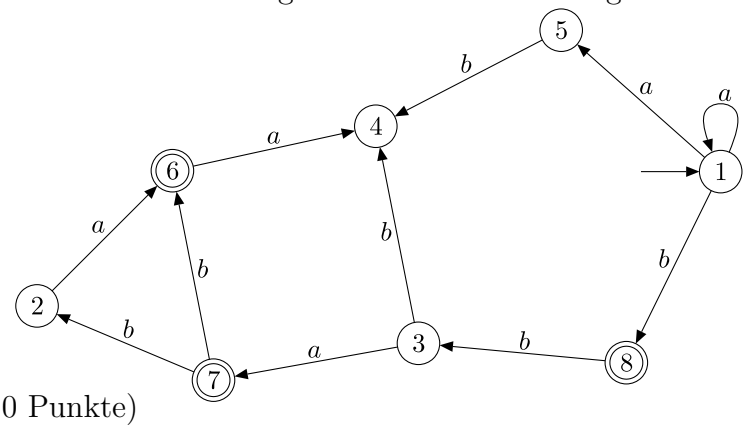
Hier bezeichnet $\mathrm{bin}(w)$ die Zahl, deren Binärdarstellung w ist. Zum Beispiel ist $\mathrm{bin}(0101) = 5$.

Aufgabe T14

Konstruieren Sie den minimalen deterministischen Automaten zu folgendem NFA. Führen Sie die Potenzmengenkonstruktion aus und geben Sie die Berechnungsschritte der Minimierung an.



Aufgabe H10 (10 Punkte)
Konstruieren Sie den minimalen deterministischen Automaten zu folgendem NFA. Führen Sie die Potenzmengenkonstruktion aus und geben Sie die Berechnungsschritte der Minimierung an.



Aufgabe H11 (10 Punkte)
Welche der folgenden Sprachen sind regulär und welche nicht? Beweisen Sie Ihre Behauptung.

1. $L_1 = \{ a^n \mid \sqrt{n} \in \mathbf{N},\, n > 100 \}$
2. $L_2 = L_1^*$

Die zweite Aufgabe ist nicht einfach und kann nicht unmittelbar mit dem Stoff dieser Vorlesung gelöst werden. Sie brauchen dafür auch Kenntnisse aus anderen Vorlesungen, die Sie wahrscheinlich schon gehört haben. Es ist eine kleine Forschungsaufgabe, durch die Sie sich vielleicht durchbeißen müssen, aber geben Sie nicht auf: Wenn Sie es geschafft haben, steht einer Karriere als Wissenschaftlerin oder Wissenschaftler nichts mehr im Weg.

Aufgabe H12 (10 Punkte)
In der Vorlesung haben wir gesehen wie man testen kann, ob ein Wort zur Sprache eines regulären Ausdrucks gehört, der dafür in einen NFA verwandelt wird. Nun betrachten wir konkrete Implementierungen und vergleichen sie.
Es gibt viele Implementierungen, die die Erkenntnisse aus FoSAP ignorieren und direkt auf dem regulären Ausdruck arbeiten. Das sind unter anderem Implementierungen, die es schon seit Jahrzehnten gibt, an denen viel optimiert worden ist und die auch viel genutzt werden. Dazu gehören die Standardbibliotheken vieler Programmiersprachen. In dieser Aufgabe wollen wir herausfinden, ob das eine gute Idee ist.
Aufgabe: Wir betrachten den regulären Ausdruck R in Unixschreibweise `(.*?,){13}z` und Eingabewörter $w_i = \texttt{1,2,3,4,5,6,7,8,9,10,11,12}(\texttt{,1})^i$ für $1 \leq i \leq 20$. Zum Beispiel ist $w_3 =$ `1,2,3,4,5,6,7,8,9,10,11,12,1,1,1` . Wir wollen nun herausfinden, ob w_i von R erkannt wird (für $1 \leq i \leq 20$). Dies wird oft als *match* bezeichnet.
a) Wählen Sie jeweils mindestens eine Implementierung aus Kategorie 1 und 2 und finden Sie heraus, wie lange es braucht um jedes Eingabewort w_i auf Zugehörigkeit zu R zu testen. Geben Sie ihre Messwerte an und beschreiben Sie, was zu erkennen ist.
b) Woher kommen diese großen Unterschiede in der Laufzeit? Sind Sie überrascht? Formulieren Sie erst eine Vermutung. Recherchieren Sie anschließend und fassen Sie Ergebnisse zusammen. Können Sie einen Fall finden, wo diese Laufzeit sich verheerend in der Realität ausgewirkt hat?
c) Testen Sie nun Ihre NFA-Implementierung von letzter Woche mit derselben Eingabe. Vergleichen Sie das Resultat mit den vorherigen. Ist der Vergleich zu den anderen Implementierungen fair?
Geben Sie auch Ihren genutzten Quellcode ab!
Kategorie 1: Programme bzw. Standardbibliotheken von Sprachen, die das Problem mit Automaten lösen: egrep, Rust, Go, RE2 von Google.
Kategorie 2: Standardbibliotheken von Sprachen, die das Problem nicht mit Automaten lösen: Perl, Java, Python, C++.