Prof. Dr. Jasmin Blanchette
Jannis Limperg

Ludwig-Maximilians-Universität München
Institut für Informatik
Abgabe bis 10. Mai 2023, 10:00 Uhr

Übung 3 zur Vorlesung
Formale Sprachen und Komplexität

FSK3-1 ***Konstruktion von NFAs*** (2 Punkte)

a) Geben Sie den Zustandsgraph eines NFA A mit Eingabealphabet $\{a, b, c, d\}$ an, der genau alle Wörter w akzeptiert, für die gilt:

- w ist von der Form a^{4i} für ein $i \in \mathbb{N}_{>0}$ **oder**
- w fängt mit a an und endet mit einem b.

D.h. die von A akzeptierte Sprache kann geschrieben werden als

$$L(A) = \{w \in \{a\}^* \mid \#_a(w) = 4i, i \in \mathbb{N}_{>0}\} \cup \{awb \mid w \in \{a, b, c, d\}^*\}$$

b) Sei $\Sigma = \{1, 2, 3\}$ und

$$L = \{w \mid i \in \Sigma, w \in \Sigma^* \text{ und } \#_i(w) = i\}.$$

Das heißt die Sprache L enthält genau die Wörter w, für die gilt: Es gibt eine Zahl $i \in \{1, 2, 3\}$ sodass das Wort w das Symbol i genau i-mal enthält.

Z.B. gilt $212323 \in L$, da das Wort 212323 genau 1-mal das Symbol 1 enthält. Ebenso gilt $2311233 \in L$, da das Wort 2311233 genau 2-mal das Symbol 2 enthält. Hingegen ist $112223 \notin L$.

Geben Sie den Zustandsgraph eines NFA B an mit $L(B) = L$.

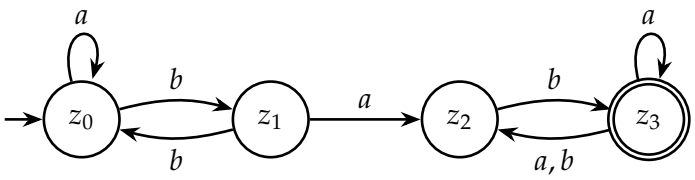
c) Sei M ein beliebiger NFA über einem beliebigen Alphabet Σ und sei $a \notin \Sigma$. Konstruieren Sie allgemein daraus einen NFA N über $\Sigma \cup \{a\}$ mit

$$L(N) = \{wa \mid w \in L(M)\}$$

Anders gesagt: Der Automat N soll ein Wort genau dann akzeptieren, wenn das Wort von der Form wa ist und w vom Automaten M akzeptiert wird.

FSK3-2 ***Läufe und Potenzmengenkonstruktion*** (2 Punkte)

Für diese Aufgabe betrachten wir folgenden NFA C über dem Alphabet $\{a, b\}$:



a) Geben Sie einen akzeptierenden Lauf des NFA C auf dem Wort $w = aaabbbabab$ an.

b) Berechnen Sie zum NFA C einen äquivalenten DFA D mit der Potenzmengenkonstruktion. Geben Sie nur den Zustandsgraphen des vom Startzustand erreichbaren Teils des Automaten an. Vergessen Sie nicht, den Startzustand und die Endzustände zu markieren.

c) Geben Sie den Lauf des DFA D auf dem Wort w an. Ist der Lauf akzeptierend?

Welcher Zusammenhang besteht zwischen dem von Ihnen in der Teilaufgabe a) gefundenen Lauf des Automaten C und dem in dieser Teilaufgabe gefundenen Lauf des Automaten D auf dem Wort w?

FSK3-3 ***Tokenizer*** (0 Punkte)

Ein Einsatzgebiet für endliche Automaten sind Tokenizer. Diese werden verwendet, um den Quelltext einer Programmiersprache in syntaktische Einheiten (Tokens) zu zerlegen. Ein Token ist beispielsweise ein Schlüsselwort, ein Bezeichner, ein Operator, etc.

Zum Beispiel wird das Programm

```
if(x==y){z=x};
```

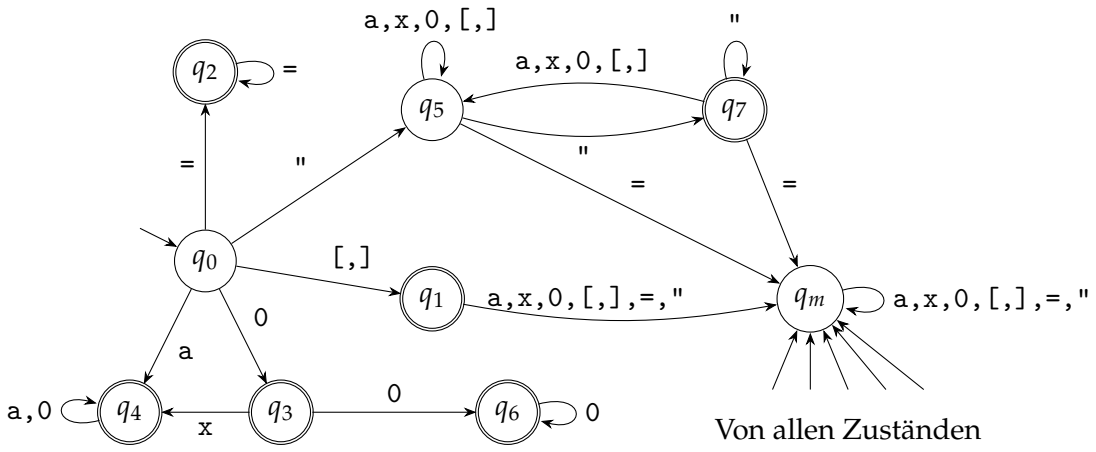
zerlegt in

„`if`“ „`(`“ „`x`“ „`==`“ „`y`“ „`)`“ „`{`“ „`z`“ „`=`“ „`x`“ „`}`“ „`;`“

In dieser Aufgabe erstellen wir einen Tokenizer, indem wir die möglichen Tokens als reguläre Sprache auffassen.

a) Um alle Schritte sinnvoll per Hand rechnen zu können, arbeiten wir mit einem reduzierten Alphabet (`[]` statt `()` oder `{}`, weniger Buchstaben aus dem Alphabet, nur eine Ziffer, ...):
$\Sigma = \{$`a,x,0,[,],=,"`$\}$

Die Sprache der möglichen Tokens ist als DFA A gegeben:



Um das erste Token aus einem String zu identifizieren, wird A vom Anfang des Strings aus laufen gelassen. Wenn der Lauf nie in einen Endzustand kommt, meldet der Tokenizer einen Fehler. Ansonsten wird die *letzte* Position, in welcher der Automat in einem Endzustand war, als Token-Ende genommen.

Zum Beispiel ist bei Eingabe `==aa[` der Lauf $q_0 \xrightarrow{=} q_2 \xrightarrow{=} q_2 \xrightarrow{a} q_m \xrightarrow{a} q_m \xrightarrow{[} q_m$. Da q_2 akzeptierend ist (aber q_m nicht), ist das erkannte Token `==`.

Notieren Sie bei folgenden Strings die Zustände, die A bei Verarbeitung dieser Strings annehmen wird (die Läufe) und geben Sie je die Ausgabe des Tokenizers an. Bezüglich der Ausgabe reicht es, sofern der Tokenizer keinen Fehler zurückgibt, nur das erste erkannte Token anzugeben.

- `aa==a`
- `a[0]`
- `a[[[[]`
- `"a[0]"ax"`
- `"a=[0]"ax"`
- `"a[0]"a=x"`

b) Bestimmen Sie asymptotisch (in O-Notation), wie viele Schritte der Tokenizer-Automat braucht, um ein Token aus einem String der Länge n zu extrahieren.

c) Um mehrere Tokens zu extrahieren, wird das gefundene Token von dem String entfernt und wieder von vorne ein Token gesucht. Wenn der verbleibende String leer ist, ist der Tokenizer fertig.

Beispiel: Bei der oben genannten Eingabe `==aa[` mit dem ersten Token `==`, ist der Reststring nach dem Entfernen `aa[`, das zweite Token dann also `aa`.

Zerlegen Sie mit diesem Algorithmus den String `a="ax0"aa[0]=a` in alle Tokens.

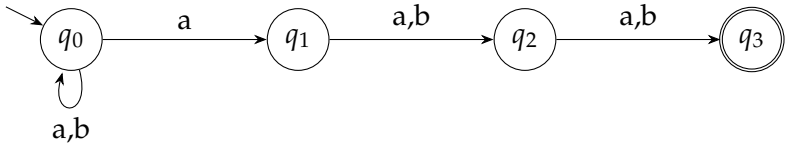
d) Tatsächlich müssen wir den String nicht verändern, sondern, wenn ein Token gefunden wurde, nur den Automat an der nächsten Position im String starten. Wir ‚kürzen' den String also in $O(1)$.

Wie viele Schritte brauchen wir dann asymptotisch, um alle Tokens aus einem String der Länge n zu finden? (Hinweis: Es ist nicht $O(n)$. Man könnte das Verfahren aber optimieren, um eine Laufzeit von $O(n)$ zu erreichen.)

FSK3-4 ***Umgedrehte Sprache*** (0 Punkte)

Sei T die Funktion, die aus einem NFA $A = (Z, \Sigma, \delta, S, E)$ einen NFA $T(A) = (Z, \Sigma, \delta', E, S)$ erzeugt, wobei $p \in \delta'(q, a) \iff q \in \delta(p, a)$.

a) Berechnen Sie den Zustandsgraph des Automaten $B = T(A)$ für folgenden Automaten A:



b) Geben Sie den Zustandsgraph eines DFA C mit $L(B) = L(C)$ an. (Sie dürfen die Potenzmengenkonstruktion nutzen, müssen aber nicht.)

c) Zeigen Sie: Für jeden NFA A ist $L(T(A)) = \{w \mid \overline{w} \in L(A)\}$. Dabei steht $\overline{w}$ wie in der Vorlesung für das rückwärts gelesene Wort w.

Hinweis: Es kann hilfreich sein, zuerst $L(T(A)) \subseteq \{w \mid \overline{w} \in L(A)\}$ und dann $\{w \mid \overline{w} \in L(A)\} \subseteq L(T(A))$ zu zeigen.