Formale Sprachen

Marco Haupt, KA-TINF22B1, Übungsblatt #3

Es empfiehlt sich bei allen Aufgaben zu regulären Ausdrücken einen (online) Tester zu verwenden. Zum Beispiel: https://regex101.com/

Denken Sie sich dazu ggf. eigene Zeichenketten aus und überprüfen Sie, ob Ihr regulärer Ausdruck die gewünschten Treffer erzielt. Falls nicht, versuchen Sie nachzuvollziehen, warum ihre Vermutung falsch war, und versuchen Sie es erneut, indem Sie den Regex anpassen. Schauen Sie nur in die Musterlösung, wenn Sie gar nicht mehr weiterwissen oder um Ihre Lösung zu verifizieren.

Aufgabe 3.1

Geben Sie für jede der folgenden formalen Sprachen einen regulären Ausdruck an, der sie beschreibt:

a) $L = \{w \in \{0,1\}^* \mid \text{in } w \text{ kommt mindestens eine } 0 \text{ und mindestens eine } 1 \text{ vor}\}$
b) $L = \{w \in \{0,1\}^* \mid \text{in } w \text{ kommt nicht die Zeichenfolge } 001 \text{ vor}\}$
c) Die Sprache aller Wörter über dem Alphabet $\{a, b, c\}$, die mindestens ein a und mindestens ein b enthalten.
d) Die Sprache aller Wörter über dem Alphabet $\{0,1\}$, deren drittes Symbol von rechts eine 1 ist.
e) Die Sprache aller Wörter über dem Alphabet $\{0,1\}$, in denen alle Paare von Nullen vor allen Paaren von Einsen stehen.
f) Die Sprache aller Wörter über dem Alphabet $\{0,1\}$, in denen die Anzahl der Nullen durch 5 teilbar ist.

Aufgabe 3.2

Beschreiben Sie, welche Zeichenketten die folgenden Regexe matchen:

1. `www.ibm.com`
2. `[01]?[0-9]:[0-9][0-9](am|pm)`
   Was könnte man hier optimieren? Und wie?
3. `[()]`
4. `([a-z])\1`
5. `[.*].*`

Aufgabe 3.3

Es sei das Wort $aaabab$ zu Grunde gelegt. Geben Sie für jeden der folgenden Regexe an, welches Teilwort gematcht wird.

1. `a.*`
2. `.a*`
3. `.*a`
4. `b.*`
5. `.b*`
6. `.*b`

3.1)
a) ^(?=.*0)(?=.*1)[01]*$
b) ^(?!.*001)[01]*$
c) ^(?=.*a)(?=.*b)[abc]*$
d) ^(?:[01]{2}1|1)[01]*$
e) ^(0*10*1)*0*$
f) ^(1*01*01*01*01*0)*1*$

3.2)
1.) Genau die Zeichenkette www.ibm.com
2.) Uhrzeit hh:mm am/pm (12 h Zeit)
3.) entweder offene oder geschlossene Klammer
4.) 2 kleinbuchstaben hintereinander (zwischen a und z)
5.) ein beliebiges Zeichen, dann ein Punkt, dann viele weitere Zeichen

3.3)
1.) aaababb
2.) aaa
3.) aaaba
4.) babb
5.) a
6.) aaababb

3.4)
1.) [0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}
2.) [a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}
3.) \+\d{1,4}\s?(\d{1,}-?){1,}(\d{1,})
4.) (\w)\1

3.5)
G# = (N, T, S', P#)
N = N ∪ {S'}
S' ist ein neues Startsymbol
P# = P ∪ {S' -> SS', S' -> ε}

3.6)
G = (N, T, S, P)
G$ = (N$, T, S$, P$)
G% = (N%, T, S%, P%)

G$ = (N$ = N, T, S$, P$ = P ∪ {S$ -> ε})
G% = (N% = N, T, S%, P% = P ∪ {S% -> ε})

G = (N, T, S, P = P ∪ {S -> S$S%})

3.7)
G = (N, T, S, P)
G' = (N', T', S', P')

G = (N, T, S, P)
G' = (N', T', S', P')

G'' = (N'' = N ∪ N', T'' = T ∪ T', S'' = S''', P'' = P ∪ P' ∪ {S''' -> SS', S''' -> SS''})

Hierbei ist S''' ein neues Startsymbol.

3.9)
a) anzahl a und b gleich
b) alle Palindrome aus a und b
c) wie bei a

3.10)
a) R = (abc){3k}, wobei $k \geq 0$
b) regulär

Aufgabe 3.4

Kopieren Sie den Inhalt von RFC 1918 in den Bereich TEST STRING bei https://regex101.com/. Geben Sie nun jeweils einen Regex an, der alle

- IP-Adressen,
- E-Mail-Adressen,
- Telefonnummern,
- und Worte mit Doppelkonsonanten

im Text findet. Versuchen Sie Ihren Regex bei maximaler Präzision so kurz wie möglich zu halten.

Aufgabe 3.5

Es sei $G = (N, T, X_0, P)$ eine Typ-3-Grammatik. Konstruieren Sie eine Typ-3-Grammatik G_S, so dass gilt: $L(G_S) = L(G)^*$.

Aufgabe 3.6

Es seien $G_1 = (N_1, T_1, S_1, P_1)$ und $G_2 = (N_2, T_2, S_2, P_2)$ Typ-3-Grammatiken. Konstruieren Sie eine Typ-3-Grammatik G, sodass gilt: $L(G) = L(G_1) \cdot L(G_2)$.

Aufgabe 3.7

Es seien $G_1 = (N_1, T_1, S_1, P_1)$ und $G_2 = (N_2, T_2, S_2, P_2)$ Typ-3-Grammatiken. Konstruieren Sie eine Typ-3-Grammatik G, sodass gilt: $L(G) = L(G_1) \cup L(G_2)$. Nehmen Sie an, dass $N_1 \cap N_2 = \emptyset$.

Aufgabe 3.8

Seien L_1 und L_2 zwei beliebige Sprachen über $\{a, b\}$. Zeigen Sie: ${L_1}^* \cap {L_2}^* \neq \{\}$

Aufgabe 3.9

Welche formalen Sprachen erzeugen die folgenden Grammatiken?

1. $G_1 = (\{S, A, B\}, \{a, b\}, S, \{S \to aA, A \to Sa, S \to bB, B \to Sb, S \to \varepsilon\})$
2. $G_1 = (\{X\}, \{a, b\}, X, \{X \to XX, X \to abXba, X \to \varepsilon\})$
3. $G_1 = (\{X\}, \{a, b\}, X, \{X \to XX, X \to aXb, X \to bXa, X \to \varepsilon\})$

Tipp: Erzeugt ein paar Wörter der Sprache (vorzugsweise die kürzesten) und versucht darin ein Schema zu erkennen. Betrachtet dabei unbedingt alle Produktionen der Sprache!

Aufgabe 3.10

Sei L eine Sprache über dem Alphabet $T = \{a, b\}$, sodass L alle Worte $w \in T^*$ beschreibt, für die $|w|$ durch 3 teilbar ist.

1. Geben Sie einen regulären Ausdruck R an, der die Sprache $L(G)$ beschreibt.
2. Von welchem Sprachtyp ist $L(G)$?

Aufgabe 3.11

1. Geben Sie eine Grammatik G mit Terminalzeichenalphabet $T = \{a, b, c\}$ und ein Wort $w \in T^+$ an, das in G mehrere Linksableitungen hat.
2. Wie viele Linksableitungen gibt es für w in Ihrer Grammatik?
3. Wie viele Rechtsableitungen gibt es für w in Ihrer Grammatik?
4. Wie ist allgemein der Zusammenhang zwischen der Anzahl der Links- und der Anzahl der Rechtsableitungen eines Wortes in einer Grammatik?
5. Begründen Sie Ihre Aussage aus Punkt 4.
6. Gibt es eine Grammatik, bei der ein Wort unendlich viele Linksableitungen hat? Wenn ja, geben Sie ein Beispiel an; wenn nein, geben Sie eine Begründung an.

Aufgabe 3.12

Gegeben sei die formale Sprache $L = (\{a\}\{ab\}^*\{b\})^*$ und $w \in L$.

1. Was können Sie über die Anzahl der a und b in w sagen?
2. Was können Sie über die Anzahl der a und b in einem Anfangsstück von w sagen?
3. Beschreiben Sie umgangssprachlich aber präzise, welche Wörter L enthält.

Aufgabe 3.13

Stellen Sie sich vor in Ihrer Abteilung gäbe es folgende Code-Konvention:

Parameternamen bestehen ausschließlich aus Buchstaben des lateinischen Alphabets ohne Sonderzeichen. Sie beginnen mit den Präfixen i, o oder io, um die beabsichtigte Verwendung als Input-, Output- oder InOut-Parameter kenntlich zu machen. Danach folgt ein Großbuchstabe, gefolgt von beliebig vielen weiteren Zeichen.

Erstellen Sie einen regulären Ausdruck zur Beschreibung valider Parameternamen.

Gültige Parameternamen sind zum Beispiel: iStudents, oGrades, ioSalary

Aufgabe 3.14

Es sei $L \subseteq \{a, b\}^*$ die formale Sprache $L = \{a^k b^k \mid k \in \mathbb{N}_0\}$. Kann man L als Produkt von zwei formalen Sprachen schreiben? Falls ja, wie? Und wie nicht?

Aufgabe 3.15 (Python-Hands-On)

In der auf Moodle zur Verfügung gestellten Datei `simpsons-phone-book.xml` sind die Vornamen, Nachnamen und Telefonnummern einiger Bewohner Springfields gelistet.

Schreiben Sie ein Python-Skript (als Jupyter-Notebook oder .py-Datei) welches die Datei zeilenweise einliest und die enthaltenen Daten in folgendem Format ausgibt:

```
Telefonnummer Nachname, Vorname
```

Verwenden Sie dazu Regexes und die Methode `re.search(…)`. Ihr Regex darf nicht die Schlüsselwörter `firstname`, `lastname` oder `phonenumber` beinhalten. In der Datei `tinf20b1-exercise-2-19.ipynb` sind bereits einige Beispiele und Hilfestellungen enthalten.

3.12)
a) w hat gleich viele a und b
b) jedes Präfix in w hat gleich viele a und b
c) Paare aus a und b mit je gleich vielen a und b, beliebig wiederholbar

3.13)
(ilolio)[A-Z][A-Za-z]*

3.14) Nein, da die Anzahl der a und b voneinander abhängt

3.15)

```
import re

with open('simpsons-phone-book.xml', 'r') as file:
    data = file.readlines()

for line in data:
    match = re.search(r'<firstname>(.*?)</firstname>.*<lastname>(.*?)</lastname>.*<phonenumber>(.*?)
</phonenumber>', line)
    if match:
        firstname = match.group(1)
        lastname = match.group(2)
        phonenumber = match.group(3)
        print(f"{phonenumber} {lastname}, {firstname}")
```

Übung 3.16: Matches erkennen

Geben Sie für jeden der folgenden regulären Ausdrücke an, welcher Teil des Textes vom regulären Ausdruck gematcht wird, indem Sie die entsprechenden Stellen unterstreichen. Falls es keinen Match gibt, kreuzen Sie dies an. Markieren Sie nur den ersten Match, falls es mehrere gibt.

`k*`	k m l l l k k m k k m m	☐ kein Match
`m*`	k m l l l k k m k k m m	☐ kein Match
`(lk)*mk*`	k m l l l k k m k k m m	☐ kein Match
`.*kk`	k m l l l k k m k k m m	☐ kein Match
`k(.)*l`	k m l l l k k m k k m m	☐ kein Match
`lk?l`	k m l l l k k m k k m m	☐ kein Match
`(kl)?m`	k m l l l k k m k k m m	☐ kein Match
`k*l`	k m l l l k k m k k m m	☐ kein Match
`k+l`	k m l l l k k m k k m m	☐ kein Match
`[kkm]+`	k m l l l k k m k k m m	☐ kein Match
`(kkm)+`	k m l l l k k m k k m m	☐ kein Match
`kk(..)+k`	k m l l l k k m k k m m	☐ kein Match
`(k|l)m(k|l)`	k m l l l k k m k k m m	☐ kein Match
`(k|l)m\1`	k m l l l k k m k k m m	☐ kein Match
`(km).*\1`	k m l l l k k m k k m m	☐ kein Match
`([kl]+)\1`	k m l l l k k m k k m m	☐ kein Match
`l{1,5}`	k m l l l k k m k k m m	☐ kein Match
`l{1,2}`	k m l l l k k m k k m m	☐ kein Match
`l{1,2}k`	k m l l l k k m k k m m	☐ kein Match
`l{2,}k`	k m l l l k k m k k m m	☐ kein Match
`[^k]+`	k m l l l k k m k k m m	☐ kein Match
`[^k]+$`	k m l l l k k m k k m m	☐ kein Match
`l+`	k m l l l k k m k k m m	☐ kein Match
`^l+`	k m l l l k k m k k m m	☐ kein Match
`kkm*`	k m l l l k k m k k m m	☐ kein Match
`kkm*$`	k m l l l k k m k k m m	☐ kein Match
`.`	V e n i . V i d i . V i c i .	☐ kein Match
`.+`	V e n i . V i d i . V i c i .	☐ kein Match
`.+$`	V e n i . V i d i . V i c i .	☐ kein Match
`\.+`	V e n i . V i d i . V i c i .	☐ kein Match

[.]+	V e n i . V i d i . V i c i .	☐ kein Match
[.]+$	V e n i . V i d i . V i c i .	☐ kein Match