Theoretische Informatik
Übung "Reguläre Ausdrücke"

Prof. Dr. Jürgen Brauer

Aufgabe 1 - Reguläre Grammatik? Reguläre Ausdrücke

1.1) Gegeben sei die Grammatik
$G_1 = (\{S, X, Y, Z\}, \{a, b, c\}, \{S \to aX, X \to bY, Y \to cZ, Z \to \varepsilon | Y\}, S)$.
Welche Sprache wird von dieser Grammatik erzeugt?
Handelt es sich um eine reguläre Grammatik?
Wenn nein, wandeln Sie die Grammatik in eine äquivalente reguläre Grammatik um!
Nennen Sie einen regulären Ausdruck r, der die gleiche Sprache erzeugt, d.h. $\mathcal{L}(r) = L(G_1)$

Erzeugte Sprache: $L(G_1) = \{abc, abcc, abccc, \dots\} = \{abc^n : n \in \mathbb{N}\}$
Keine reguläre Grammatik wegen der Regel $Z \to Y$
Umwandlung: $P = \{S \to aX, X \to bY, Y \to cY | cZ, Z \to \varepsilon\}$
Regulärer Ausdruck: $r = abcc^*$

1.2) Gegeben sei die Grammatik
$G_2 = (\{S, X, Y, Z\}, \{a, b, c, d\}, \{S \to XYZ, X \to aX | a, Y \to b | c, Z \to dZ | d\}, S)$.
Welche Sprache wird von dieser Grammatik erzeugt?
Handelt es sich um eine reguläre Grammatik?
Nennen Sie einen regulären Ausdruck s, der die gleiche Sprache erzeugt, d.h. $\mathcal{L}(s) = L(G_2)$

Erzeugte Sprache: $L(G_2) = \{abd, acd, aabd, aacd, abdd, acdd, \dots\} =$
$\{a^m b^n c^o d^p : m, p \in \mathbb{N},\ n, o \in \{0, 1\},\ n + o = 1\}$
Keine reguläre Grammatik wegen z.B. der Regel $S \to XYZ$ oder $X \to a$
Regulärer Ausdruck: $s = aa^*(b|c)dd^*$

1.3) Gegeben sei der reguläre Ausdruck $t = (a|b)^*c$.
Geben Sie eine Grammatik G_3 an, die die gleiche Sprache erzeugt, d.h. $\mathcal{L}(t) = L(G_3)$

$G_3 = \{\{S, X\}, \{a, b, c\}, \{S \to Xc, X \to aX|bX|\varepsilon\}, S\}$.

Aufgabe 2 - Reguläre Ausdrücke in C++

Seit dem C++ Standard C++11 gibt es zur Realisierung von regulären Ausdrücken die Klasse std::regex.

2.1) Nutzen Sie diese Klasse und schreiben Sie ein Beispielprogramm in C++, das mit Hilfe von verschiedenen regulären Ausdrücken folgende Zahleneingaben erlaubt bzw. zurückweist:

```
1. Please enter a single digit,
e.g. 5
9
Input ok!

2. Please enter a single or multiple digits (without a sign),
e.g. 559
1005
Input ok!

3. Please enter a single or multiple digits (with or without a minus sign -),
e.g. -559
-1005
Input ok!

4. Please enter a single or multiple digits (with or without a plus sign +),
e.g. +559
+3209
Input ok!

4. Please enter a single or multiple digits (with or without a minus - or plus sign +),
e.g. -903
-9000
Input ok!

5. Please enter a real number (with or without a minus - or plus sign +),
e.g. -3.14159
2.71828
Input ok!

1. Please enter a single digit,
e.g. 5
```

```
200
Input wrong!

2. Please enter a single or multiple digits (without a sign),
e.g. 559
--399
Input wrong!

3. Please enter a single or multiple digits (with or without a minus sign -),
e.g. -559
+9999
Input wrong!

4. Please enter a single or multiple digits (with or without a plus sign +),
e.g. +559
333+33
Input wrong!

4. Please enter a single or multiple digits (with or without a minus - or plus sign +),
e.g. -903
-9.9
Input wrong!

5. Please enter a real number (with or without a minus - or plus sign +),
e.g. -3.14159
192.199.200.201
Input wrong!
```

2.2) Erstellen Sie einen regulärern Ausdruck, der überprüft, ob der Benutzer eine gültige eMail-Adresse eingegeben hat:

```
Enter an email address:
hello
Input wrong!

Enter an email address:
hello@
Input wrong!

Enter an email address:
hello@somewhere
Input wrong!
```

```
Enter an email address:
hello@somewhere.com
Input ok!

Enter an email address:
@somewhere.com
Input wrong!

Enter an email address:
,@shomewhere.com
Input wrong!

Enter an email address:
hello.all@somewhere.com
Input ok!

Enter an email address:
hello.all@somewhere,com
Input wrong!

Enter an email address:
hello_@somewhere.com
Input wrong!

Enter an email address:
hello:@somwhere.com
Input wrong!

Enter an email address:
.all@somewhere.com
Input wrong!
```