Operationen auf regulären Ausdrücken

Aufgabenstellung

Auf dem aktuellen Übungsblatt finden Sie die folgende rekursiv definierte Prozedur, die überprüft, ob ein regulärer Ausdruck die leere Sprache erkennt:

$$\begin{aligned} \text{is\_empty}(\emptyset) &= \text{true} & \text{is\_empty}(r\,s) &= \text{is\_empty}(r) \vee \text{is\_empty}(s) \\ \text{is\_empty}(\varepsilon) &= \text{false} & \text{is\_empty}(r \mid s) &= \text{is\_empty}(r) \wedge \text{is\_empty}(s) \\ \text{is\_empty}(a) &= \text{false} & \text{is\_empty}(r^*) &= \text{false} \end{aligned}$$

Hierbei bezeichnen r und s reguläre Ausdrücke über irgendeinem Alphabet Σ und $a \in \Sigma$ ist ein beliebiges Symbol.

Analog dazu definieren wir nun eine Prozedur *is_trivial*, die entscheidet, ob $L(r) \subset \{\varepsilon\}$ gilt, also ob die Sprache, die der reguläre Ausdruck erkennt, entweder die leere Sprache $\emptyset$ ist oder die Sprache $\{\varepsilon\}$:

$$\begin{aligned} \text{is\_trivial}(\emptyset) &= \text{true} & \text{is\_trivial}(r\,s) &= \text{is\_empty}(r\,s) \vee (\text{is\_trivial}(r) \wedge \text{is\_trivial}(s)) \\ \text{is\_trivial}(\varepsilon) &= \text{true} & \text{is\_trivial}(r \mid s) &= \text{is\_trivial}(r) \wedge \text{is\_trivial}(s) \\ \text{is\_trivial}(a) &= \text{false} & \text{is\_trivial}(r^*) &= \text{is\_trivial}(r) \end{aligned}$$

Machen Sie sich mit der Java-Implementierung von regulären Ausdrücken vertraut. Diese besteht aus einer abstrakten Klasse `Regex` und 6 Unterklassen – eine für jeden der 6 obigen Fälle. Die obige Prozedur is_empty ist dort bereits implementiert.

- Implementieren Sie die oben skizzierte *is_trivial*-Prozedur. Vervollständigen Sie dafür die Methodenrümpfe der `isTrivial`-Methode in den 6 Unterklassen von Regex. Sie können sich dabei an der Implementierung von `isEmpty` orientieren.
- Überlegen Sie sich auf Papier eine ähnliche rekursive Prozedur, die entscheidet, ob ein gegebener regulärer Ausdruck eine endliche Sprache erkennt.
- Implementieren Sie eine Methode `enumerateWords`, die die Menge *aller* Wörter zurückgibt, die von dem regulären Ausdruck erkannt wird sofern diese endlich ist bzw. `null` wenn sie unendlich ist. Auch hier bietet sich das gleiche rekursives Schema wie bei den anderen beiden Prozeduren an. Hierbei dürfen (und sollten) Sie die bestehenden Methoden `isEmpty` und `isTrivial` verwendet.

Hinweis: Wenn Sie eines unserer Templates verwenden, wird die Ein- und Ausgabe komplett vom Template übernommen. Sie müssen nur die oben beschriebenen Methoden implementieren. Die folgende Beschreibung des Ein-/Ausgabeformats dient nur dem Verständnis, was die Beispiel-Testdaten bedeuten bzw. falls Sie selbst sich noch eigene Tests ausdenken wollen.

Eingabe

Eine Zeile, die entweder das Wort `Language` enthält oder das wort `Trivial`.
Dann: beliebig viele Zeilen mit je einem regulären Ausdruck pro Zeile.
Abschließend die Zeile `END`.

Das Format der regulären Ausdrücke lässt sich am einfachsten an folgendem Beispiel ablesen: `a(b|c)*(ε|∅*)`. Der Einfachheit halber können Sie $\emptyset$ auch als `{}` schreiben und ein ε als `()` oder ganz weglassen (z.B. `a(|b)`).

Ausgabe

Im `Trivial`-Modus: Für jeden Regex die Ausgabe `yes` oder `no`.
Im `Language`-Modus: Für jeden Regex entweder die gesamte Sprache im Format `{aa, ab, bb}` (in aufsteigend lexikographischer Ordnung) oder `infinite`. Das leere Wort ist als `ε` zu schreiben.

Beispiele

Hinweis: In den folgenden Beispiel-Ein-/Ausgaben fehlen leider aus technischen Gründen. einige Unicode-Zeichen. Bitte ignorieren Sie diese Beispiele und verwenden Sie stattdessen die Test-Ein-/Ausgaben aus der .tar.gz-Datei von der Webseite.

Sample Input 1

```
Trivial



a
a|b
|
ab
a
a
*
a(b|)*
a((b|c)*(a|))*
(b|a)(cc|d|)
(|(ab)*)*
a*b*
END
```

Sample Output 1

```
yes
yes
no
no
yes
no
no
yes
yes
no
no
no
yes
no
```

Sample Input 2

```
Language



a
a|b
|
ab
a
a
*
a(b|)*
a((b|c)*(a|))*
(b|a)(cc|d|)
(|(ab)*)*
a*b*
END
```

Sample Output 2

```
{}
{}
{a}
{a, b}
{}
{ab}
{a}
{}
{}
{a}
infinite
{a, acc, ad, b, bcc, bd}
{}
infinite
```