

Regulärer Ausdruck zu ϵ -NFA

Aufgabenstellung

In dieser Aufgabe sollen Sie die Konstruktion aus Satz 3.19, die auch in Übungsaufgabe 3.4 noch einmal genauer beschrieben wird, implementieren.

Machen Sie sich zunächst 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. Vervollständigen Sie dann den Methodenrumpf `EpsilonNFA.toEpsilonNFA(Set<Character> alphabet)` in den Unterklassen von `Regex`.

Halten Sie sich dabei *exakt* an das Schema aus der Vorlesung. Die Namen der Zustände sind dabei egal. Für die Basisfälle $\emptyset/\epsilon/a$ verwenden Sie bitte die offensichtlichen ϵ -NFAs mit jeweils 1/1/2 Zuständen und 0/0/1 Transitionen.

Hinweis: Wenn Sie eines unserer Templates verwenden, wird die Ein- und Ausgabe komplett vom Template übernommen. Sie müssen nur die oben beschriebene Methode 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 mit den Zeichen des Alphabets, getrennt durch “;”. Dann eine Zeile mit einem regulären Ausdruck. Das Format der regulären Ausdrücke lässt sich am einfachsten an folgendem Beispiel ablesen: `a (b | c)* (ϵ | \emptyset^*)`. Der Einfachheit halber können Sie \emptyset auch als `{ }` schreiben und ein ϵ als `()` oder ganz weglassen (z.B. `a (| b)`).

Ausgabe

Ein ϵ -NFA in dem Format, das Sie am einfachsten den Beispiel-Ausgaben unten entnehmen. Für die Beispiel-Ausgaben stehen auch PDFs zur Verfügung zur besseren Visualisierung.

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	Sample Output 1
<code>a;b;c;d</code>	<code>EpsilonNFA Alphabet: a;b;c;d States: 0 Init: 0 Final: Transitions: END</code>
Sample Input 2	Sample Output 2
<code>a;b;c;d</code>	<code>EpsilonNFA Alphabet: a;b;c;d States: 0 Init: 0 Final: 0 Transitions: END</code>

Sample Input 3

```
a;b;c;d
a
```

Sample Output 3

```
EpsilonNFA
Alphabet: a;b;c;d
States: 0;1
Init: 0
Final: 1
Transitions:
0;a;1
END
```

Sample Input 4

```
a;b;c;d
ab
```

Sample Output 4

```
EpsilonNFA
Alphabet: a;b;c;d
States: 0;1;2;3
Init: 0
Final: 3
Transitions:
0;a;1
1;b;2
2;c;3
END
```

Sample Input 5

```
a;b;c;d
a|b
```

Sample Output 5

```
EpsilonNFA
Alphabet: a;b;c;d
States: 0;1;2;3;4
Init: 0
Final: 2;4
Transitions:
0;;1
0;;3
1;a;2
3;b;4
END
```

Sample Input 6

```
a;b;c;d
a*
```

Sample Output 6

```
EpsilonNFA
Alphabet: a;b;c;d
States: 0;1;2
Init: 0
Final: 0;2
Transitions:
0;;1
1;a;2
2;;1
END
```

Sample Input 7

```
a;b;c;d  
(a(a|b)*(c|))*
```

Sample Output 7

```
EpsilonNFA  
Alphabet: a;b;c;d  
States: 0;1;2;3;4;5;6;7;8;9;10;11;12  
Init: 0  
Final: 0;11;12  
Transitions:  
0;;1  
1;a;2  
2;;3  
3;;4  
3;;9  
4;;5  
4;;7  
5;a;6  
6;;4  
6;;9  
7;b;8  
8;;4  
8;;9  
9;;10  
9;;12  
10;c;11  
11;;1  
12;;1  
END
```