FAKULTA PRÍRODNÝCH VIED UNIVERZITY MATEJA BELA V BANSKEJ BYSTRICI

DOKUMENTÁCIA

AKCEPTOR PRE KONKRÉTNY REGULÁRNY VÝRAZ

15. novembra 2024

Jakub Kubaliak
122nAIm, 1. ročník
Formálne jazyky a automaty

Spustenie aplikácie

Odkaz na Github repozitár: https://github.com/Voltrifrodec/formalne-jazyky-a-automaty

Vytvorenie .jar súboru a spustenie aplikácie (môže vyžadovať nainštalovaný `mvn`):

```
mvn clean package
java -jar ./target/kubaliak-zadanie-1.jar
```

Manuálne zadanie vstupu:

```
java -jar ./target/kubaliak-zadanie-1.jar
```

Zadanie vstupu zo súboru:

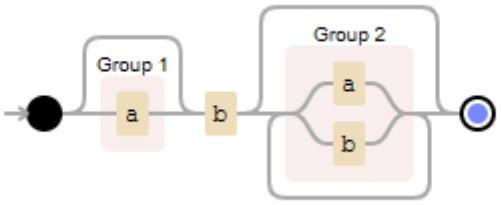
```
java -jar ./target/kubaliak-zadanie-1.jar test-retazcov.txt
```

Spustenie testov:

```
mvn test
```

Rozbor riešenia

Môj regulárny výraz bol výraz `[a]b{a|b}`:



Obrázok 1: Stavový diagram pre regulárny výraz `(a)?b(a|b)*`.

Teda, na prvej pozícii sa vyskytuje 0 alebo 1 znakov `a`, nasleduje jeden znak `b` a potom sa vyskytuje 0 alebo viac znakov `a` alebo `b`.

Akceptovanými reťazcami môže byť napríklad:

- `ab`
- `abba`
- `b`
- `bba`

Naopak neakceptovanými reťazcami môžu byť:

- `a`
- `aabb`
- `aa`
- `aab`

Pri riešení zadania som sa rozhodol pre programovací jazyk Java a využiť nástroj Maven pre testovanie výrazov.

Ako prvú vec, ktorú som urobil, bolo vytvorenie nového Maven projektu. Najprv som vytvoril konfiguračný súbor `pom.xml`, do ktorého som nastavil základné parametre pre projekt (názov, verziu Javy a pod.) a následne inicializoval vytvorený projekt príkazom `mvn clean install`. Potom som pracoval na vytvorení algoritmu, príslušných unit testov a následne pridanie podpory pre vstup z konzoly a textového súboru.

Algoritmus

Zo zadania som vyčítal, že nemôžem použiť knižnice pre regulárne výrazy. Pre riešenie je možné využiť buď jeden cyklus `while`, počas ktorého budem odoberať zo vstupného reťazca znaky podľa stavu, alebo využiť rekurzívne funkcie a tým elimininovať potrebu cyklu. V riešení som sa rozhodol pre využitie cyklu `while`.

Reťazec som overoval po častiach. Najprv som overil či nie je reťazec prázdny. Následne som overil prvý stav q_0 (reťazec obsahuje 0 alebo 1 počet znakov `a`). Po overení stavu q_0 som začal overovať druhý stav, q_1 (musí obsahovať znak `b`). Ako posledné som pomocou cyklu overoval posledný stav q_2 (reťazec obsahuje 0 alebo 1 počet znakov `a` alebo `b`).

Nasledujúci zdrojový kód obsahuje hlavnú časť programu – metódu obsahujúcu algoritmus pre akceptor.

```
1  public static String compareToRegex(String input) {
2      // Kontrola, ci vstupny retazec nie je prazdny.
3      if(input.length() == 0) {
4          return "N";
5      }
6
7      // Kontrola prveho stavu:   Ak retazec na zaciatku obsahuje znak 'a', tak musi
8      //                          dalsi znak musi byt 'b'. Inak nastava chyba. Ak obsahuje
       ,
9      //                          tak odstrani skontrolovany znak. A skontroluje ci novy
10     //                          retazec nie je prazdny a ci na zaciatku je znak 'b'
11     //                          (druhy stav).
12     if(input.charAt(0) == 'a') {
13         input = input.substring(1);
14
15         if(!input.isEmpty() && input.charAt(0) != 'b') {
16             return "N";
17         }
18     }
19     if(input.isEmpty()) {
20         return "N";
21     }
22
23     // Kontrola druheho stavu:  Retazec musi obsahovat znak 'b' (bez ohladu na vysledok
24     //                          prveho stavu). Inak nastava chyba. Ak obsahuje, tak
25     //                          odstrani skontrolovany znak.
26     if(input.charAt(0) != 'b') {
27         return "N";
28     }
29     input = input.substring(1);
30
31     // Kontrola tretieho stavu: Retazec moze obsahovat 0 a viac znakov 'a' alebo 'b'.
32     //                          Ak obsahuje iny, nastava chyba. Ak nie, skrat vstupny
33     //                          retazec o skontrolovany stav.
34     while(!input.isEmpty()) {
35         if(input.charAt(0) != 'a' && input.charAt(0) != 'b') {
36             return "N";
37         }
38         input = input.substring(1);
39     }
40
41     // Ak je vsetko v poriadku, tak sa vrati vystup 'A'.
42     return "A";
43 }
```

Zdrojový kód 1: Ukážka vytvorenej metódy `compareToRegex`.