Limbaje Formale și Automate

Tema 3 - Regex parser

Termen de predare: **17 Ianuarie 2021 - 23:55**

Responsabili temă:
Alex-Bogdan ANDREI
Mihai-Valentin DUMITRU

Data publicării: *17 Decembrie 2020*
Ultima actualizare: *05 Ianuarie 2021*

1 Cerință

In continuarea temei 2, dorim sa transformam o Expresie Regulata (ER) intr-un Automat Finit Determinist (AFD). Pentru a face asta, expresia regulata, prezentata ca sir, trebuie **parsata**, iar arborele de parsare rezultat va fi folosit pentru a genera AFN-ul echivalent. Pentru parsare si construirea arborelui de parsare veti folosi (implementa) un Automat Push-Down (APD).

Ca pasi sugerati pentru tema:

1. Pentru reprezentarea expresiilor regulate, veti implementa Tipul de Date Abstract (TDA-ul) prezentat la curs, in Python. Varianta recomandata de implementare este o ierarhie de clase.
2. PDA-ul va parsa expresia regulata si va crea o instanta de ER
3. Aplicati algoritmul de transformare ER in AFN
4. Folositi algoritmul implementat in *Tema 2* pentru a transforma AFN-ul in AFD.

Tema va fi impartita in 2 subtask-uri:

- **(1p)** Transformarea ER - AFN
- **(0.5p)** Transformarea AFN - AFD

Astfel, in cazul in care tema 2 nu a fost complet implementata (sau deloc), poate fi incercata in a 2-a parte a acestei teme.

2 Format input

Veți primi ca input un string ce reprezinta o expresie regulata. Alfabetul va trebuit construit de voi in functie de literele folosite. Acesta difera de la test la test. Expresia regulata este compusa din:

- litere mici din intervalul 'a'...'z'
- e_1e_2 sau (e_1e_2) - concatenarea a 2 subexpresii
- $e_1|e_2$ sau $(e_1|e_2)$ - reuniunea a 2 subexpresii
- $(e_1)^*$ - kleene star aplicat unei subexpresii

Exemple:

```
((bb)*(ab)*)

(((bb)*)*)|(((b)|(a))|((b)|(a)))
```

Va trebui sa tineti cont de paranteze *( )*, acestea schimband prioritatea operatiilor.

Exemplu:

```
(aa)* vs aa*
```

3 Format output

Task 1 - Transformare ER - AFN

Va trebui să formatați AFN-ul rezultat din conversie cu formatul întâlnit și la laboratorul 2:

- pe prima linie, un întreg ce reprezintă numărul de stări
- starea inițială trebuie să fie 0
- pe a doua linie, lista de stări finale, separate de câte un spațiu
- pe următoarele linii, până la sfârșitul inputului, câte o tranziție, constând într-o stare, un simbol, apoi o listă de stări următoare - elementele fiind separate de câte un spațiu

Exemplu de output NFA:

```
3
2
0 a 0
0 b 0 1
1 a 2
1 b 2
1 eps 2
```

Task 2 - Transformare AFN - AFD

Output-ul va fi acelasi din tema 2.

Va trebui să formatați AFD-ul rezultat din conversie cu formatul întâlnit și la laboratorul 2:

- pe prima linie, un întreg ce reprezintă numărul de stări

- starea inițială trebuie să fie 0
- pe a doua linie, lista de stări finale, separate de câte un spațiu
- pe următoarele linii, până la sfârșitul inputului, câte o tranziție, constând într-o stare, un simbol și o stare următoare - elementele fiind separate de câte un spațiu

Exemplu de output DFA:

```
4
2 3
0 b 1
0 a 0
1 b 2
1 a 3
2 b 2
2 a 3
3 b 1
3 a 0
```

4 Rulare

Va trebui să aveți un fișier `main.py`, care va fi rulat astfel:

```
$ python3 main.py <input-file> <output-file1> <output-file2>
```

Unde:

- `<input-file>` este calea către fișierul de input ce conține descrierea unei ER
- `<output-file>1` este calea către primul fișier de output ce conține descrierea AFN-ului rezultat din aplicarea algoritmului de conversie
- `<output-file>2` este calea către al doilea fișier de output ce conține descrierea AFD-ului rezultat din aplicarea algoritmului de conversie

În rest, sunteți liberi să structurați codul cum vreți.

5 Testare

Aveți la dispoziție toate testele folosite în procesul de testare automată. Directorul tests conține două subdirectoare:

- **in**: aici se găsesc fișierele de input, cu descrierea textuală a câte unei ER
- **ref**: aici se găsesc soluțiile de referință, cu descrierea textuală a unui AFD/AFN corespunzător fișierului cu același nume din folderul in

6 Trimitere

Tema va fi incarcata pe vmchecker2, la acest link. Submisia trebuie să constea într-o arhivă `zip` care trebuie să conțină cel puțin un fișier `main.py` direct în arhivă (nu într-un alt subdirector!). În rest, puteți include orice alte fișiere de cod necesare pentru funcționarea temei. Arhiva trebuie să conțină și un fișier README în care să descrieți pe larg implementarea voastră.

Numele arhivei trebuie să fie de forma: `Prenume_Nume_grupa_lfa_tema3.zip`.

Exemplu: `Ion_Andrei_Popescu_333CB_lfa_tema3.zip`.