Laborator LSD - Săptămâna 13

Subgrupa Suplimentară, 1.1C, 3.2C

1. Expresii regulate

Expresiile regulate sunt niște șiruri de caractere ce reprezintă șabloane sau tipare (pattern în limba engleză). Ele se construiesc pe baza unei gramatici, la fel ca și un limbaj de programare. Aceste șabloane sunt folosite pentru "recunoașterea" și manipularea unor șiruri de caractere. Analog cu expresiile aritmetice, o expresie regulată este construită prin combinarea unor expresii mai mici cu ajutorul unor operatori.

Metacaractere utilizate în construcția unei expresii regulate:

^

^ - folosim pentru a căuta un anumit tipar la începutul liniei. Exemplu: ”^abc” se va potrivi cu ”**abc**”, ”**abc**de”, ”**abc** f123c”, dar nu se va potrivi cu ”eabc” deoarece abc nu este la începutul liniei

$

$ - folosim pentru a căuta un anumit tipar la sfârșitul liniei. Exemplu: ”abc$” se va potrivi cu ”**abc**”, ” de**abc**”, ” f123c **abc**”, dar nu se va potrivi cu ”abcde” deoarece abc nu este la sfârșitul liniei

.

. - se potrivește cu orice caracter. Exemplu: ”a.c” se va potrivi cu ”**abc**”, ”**adc**”, ”**a0c**”, dar nu se va potrivi cu ”abbbc”. Atenție, deoarece nu am pus ancore de început sau de final se poate potrivi și cu ”ed**abc**de”.

[]

[] - se potrivește cu un singur caracter, dar acel caracter va fi din interiorul parantezelor drepte. Exemplu: ”nota [789]” se va potrivi cu ”**nota 9**”, ”Am luat **nota 7**”, dar nu se va potrivi cu ”nota 4” sau ”nota 789”. Un alt element util este cratima (-) care se poate folosi pentru a specifica intervale (considerând ordinea ASCII a caracterelor). De exemplu dacă vrem să scriem o expresie care se va potrivi cu orice literă mică putem scrie [a-z], dacă vrem să scriem o expresie care se va potrivi cu orice literă mică, mare sau cifră vom scrie [a-zA-Z0-9].

*

* - se potrivește de 0 sau mai multe ori. Exemplu: dacă scriem ”ab*c” se va potrivi cu ”ac”, ”abc”, ”abbbbbbbc”, ”12345abbbc54321”.

+

+ - se potrivește de 1 sau mai multe ori. Exemplu: dacă scriem ”ab+c” se va potrivi cu ”abc”, ”abbbbbbbc”, ”12345abbbc54321”, dar nu se va mai potrivi cu ”ac”.

?

? - se potrivește de 0 sau 1 ori. Exemplu: dacă scriem ”ab?c” se va potrivi cu ”ac”, ”abc”, dar nu se va potrivi cu ”abbbbbbbc”, ”12345abbbc54321”.

{,}

{,}-ne permit să specificăm un număr minim, respectiv maxim de repetiții pentru un tipar. Exemplu: [a-z]{2,5} se va potrivi cu ”ab”, ”abc”, ”abcd”, ”abcde”, dar nu se va potrivi cu ”a” sau cu ”abcdef”.

()

() - se folosesc pentru a grupa expresiile regulate. De exemplu, (ab)* se va potrivi cu ”abab”, ”ab”, ”abababababab”.

|

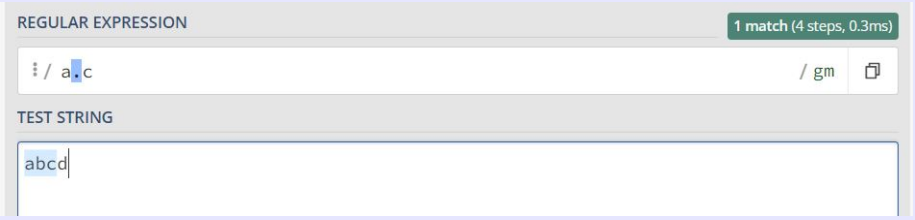
| - alternanța. De exemplu, ion (pozitiv|negativ) se va potrivi fie cu "ion pozitiv" fie cu "ion negativ".

Backslash

\ - în cazul în care vrem ca un caracter special să nu fie interpretat într-un mod special (putem deci să folosim backslash pentru escape). De exemplu, am văzut mai sus că . se potrivește cu orice caracter, dar dacă aș vrea ca acesta să nu fie interpretat (deci să pun caracterul .) voi scrie astfel: ”cartea\.”. Dacă aș vrea să scriu simbolul $ (deci să nu aibă semnificația de ancoră de final), pot proceda asemănător. Exemplu: ”[0-9]* \$”. Se va potrivi cu ”300 $”, ”Am câștigat 1000 $ ieri” ($ nu e interpretat ca și ancoră de final, pentru că am pus \ în fața lui în expresia regulată).

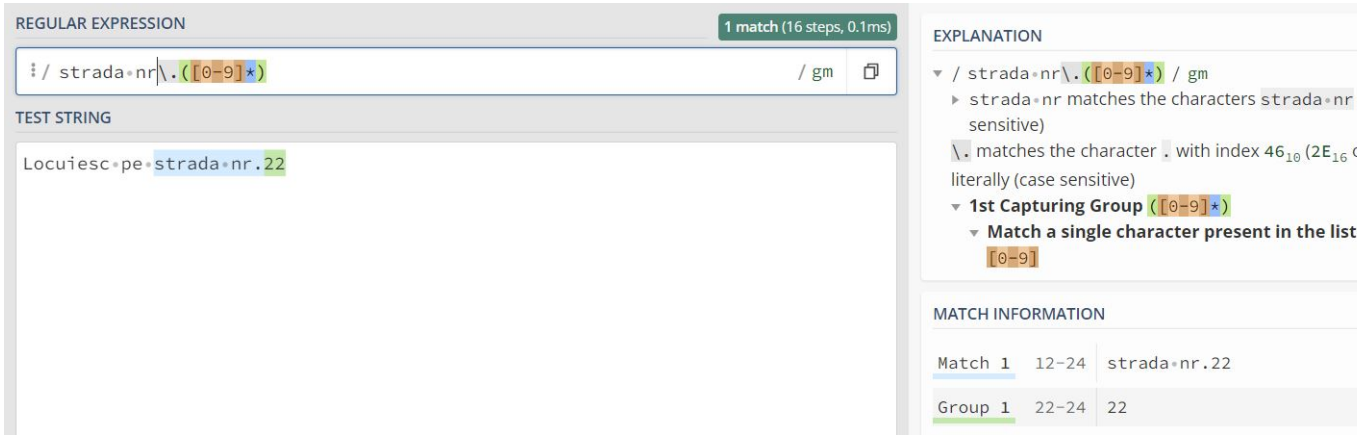
Util

La link-ul acesta puteti găsi un interpretor online pentru expresii regulate.



În câmpul cu numele REGULAR EXPRESSION introduceți expresia regulată scrisă de către voi, iar în câmpul cu numele TEST STRING puteți scrie mai multe exemple de cuvinte, fiecare pe câte un alt rând, iar cuvintele ce vor fi subliniate se vor potrivi cu expresia regulată introdusă anterior de către voi.

Un alt avantaj important al expresiilor regulate este dat de faptul că putem și extrage informații, nu doar să găsim tipare. De exemplu dacă voi defini o expresie regulată ”strada nr\.([0-9]*)” și voi folosi un grup pentru numărul străzii:



Puteți observa în partea dreaptă jos că pe lângă potrivire, în secțiunea Group 1 am extras numărul străzii cu ajutorul grupului pe care l-am definit. Puteți consulta secțiunea opțională Expresii regulate în Python pentru a vedea cum puteți face lucrul acesta în Python.

Exemplu 1

Scrieți o expresie regulată pentru cuvintele ce conțin doar caracterele 0 și 1 (pot apărea de oricâte ori și în orice combinație)

Soluție

”^(0|1)*$” sau ”^[01]*$”

Atenție

Dacă nu am fi pus ^ și $, expresia noastră regulată s-ar fi potrivit și cu șiruri de caractere de forma ”abc 01 abc”, ceea ce ar fi fost greșit deoarece acest șir conține și alte caractere în afară de 0 și 1.

Exemplu 2

Scrieți o expresie regulată pentru cuvintele ce conțin doar caracterele 0 și 1 și se termină cu 1
Soluție
”^(0|1)*1$” sau ”^[01]*1$”

Exemplu 3

Scrieți o expresie regulată pentru cuvintele ce conțin doar caracterele a și b și conțin subșirul bab
Soluție
”^(a|b)*bab(a|b)*$” sau ”^[ab]*bab[ab]*$”

Exemplu 4

Scrieți o expresie regulată pentru cuvintele ce conțin doar caracterele a și b și conțin subșirul bab
Soluție
”^(a|b)*bab(a|b)*$” sau ”^[ab]*bab[ab]*$”

Exemplu 5

Scrieți o expresie regulată pentru cuvintele ce conțin doar caracterele a și b și pot conține oricâți de b însă aceștia trebuie să fie delimitați de 2 de a.
Exemple acceptate: aba, abbbbbbba, abaaba, abbbaabaabba
Soluție
”^(ab*a)*$”

Exemplu 6

Scrieți o expresie regulată pentru cuvintele ce conțin doar caracterele a și b și au lungimea divizibilă cu 2.
Exemple acceptate: aba, abbbbbbba, abaaba, abbbaabaabba
Soluție
”^([ab][ab])*$”

2. Expresii regulate în Python (Opțional)

Python are un modul numit re pentru a lucra cu expresiile regulate. Mai multe detalii găsiți pe Campus Virtual la link-ul acesta.

3. Automate

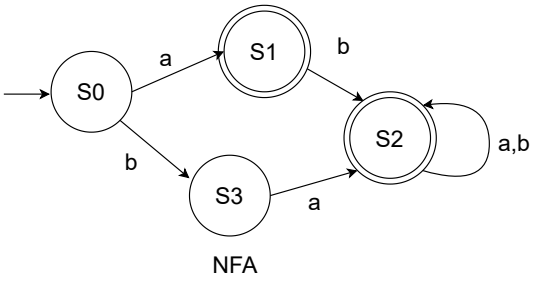
Un automat este definit prin simbolurile de intrare (formal Σ - simbolurile de intrare, ex: caractere sau apăsarea unui buton), stări (formal S - mulțimea stărilor), tranziții sau trecerea dintr-o stare in alta (formal este o funcție $\delta : S \times \Sigma \to S$), starea inițială (formal $s_0 \in S$) și mulțimea

stărilor finale (formal $F \subseteq S$, deci un automat poate avea mai multe stări finale). Formal putem defini automatul ca fiind un tuplu cu cele 5 elemente amintite mai sus $(\Sigma, S, s_0, \delta, F)$.

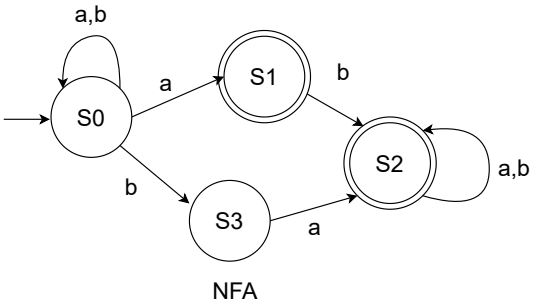
DFA = Deterministic Finite Automaton: poate fi într-o singură stare la un moment dat

NFA = Nondeterministic Finite Automaton: poate tranzita și fi în mai multe stări în același timp

Fiecare DFA este și NFA, dar nu și invers.



(a) Exemplu DFA

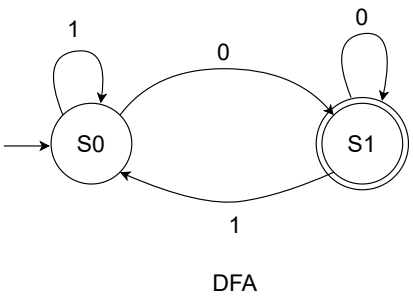


(b) Exemplu NFA

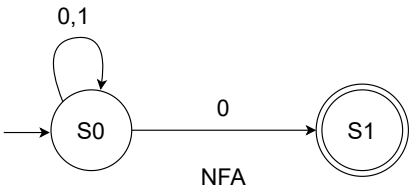
În exemplul de mai sus observăm că starea inițială este S0, iar ambele automate au ca stări finale S1 și S2. Al doilea automat este de tipul NFA deoarece din starea S0 dacă primește simbolul a poate rămâne în starea S0 sau poate trece în starea S1, deci automatul poate urma mai multe căi (spre deosebire de exemplul din prima figură). Un automat NFA va accepta dacă există o alegere care va duce într-o stare acceptoare (finală).

Exemplu 1

Considerând mulțimea de simboluri $\Sigma = \{0, 1\}$, realizați un automat care acceptă cuvintele ce se termină cu 0.



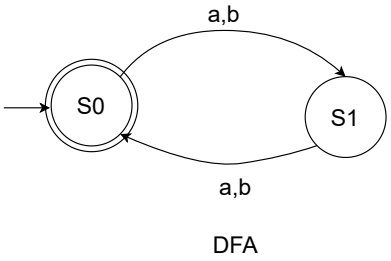
(c) DFA (ex.1)



(d) NFA (ex.1)

Exemplu 2

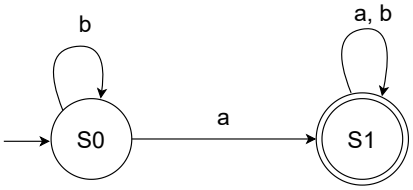
Considerând mulțimea de simboluri $\Sigma = \{a, b\}$, realizați un automat care acceptă cuvintele de lungime divizibilă cu 2.



(e) Ex.2

Exemplu 3

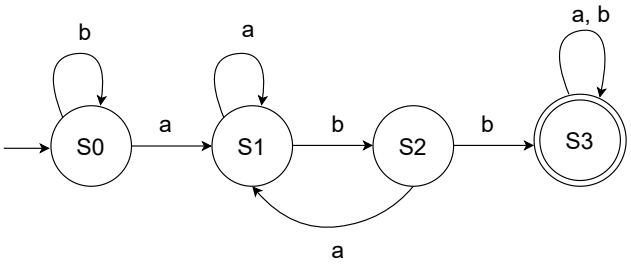
Considerând mulțimea de simboluri $\Sigma = \{a, b\}$, realizați un automat care conțin litera a (trebuie să conțină neapărat un a).



(f) Ex.3

Exemplu 4

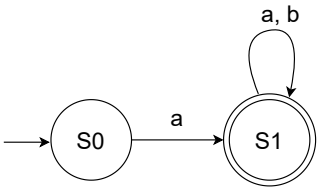
Considerând mulțimea de simboluri $\Sigma = \{a, b\}$, realizați un automat care acceptă cuvintele care conțin subșirul abb.



(g) Ex.4

Exemplu 5

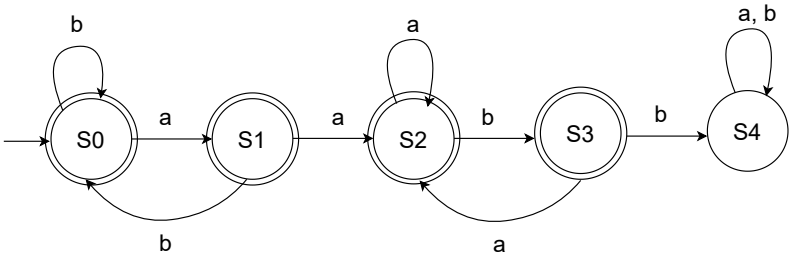
Considerând mulțimea de simboluri $\Sigma = \{a, b\}$, realizați un automat care acceptă cuvintele care încep cu a.



(h) Ex.5

Exemplu 6

Considerând mulțimea de simboluri $\Sigma = \{a, b\}$, realizați un automat care acceptă cuvintele în care dacă apare bb, atunci nu a apărut înainte aa. (deci în care nu vom accepta cuvinte ce conțin subșirul am aabb)

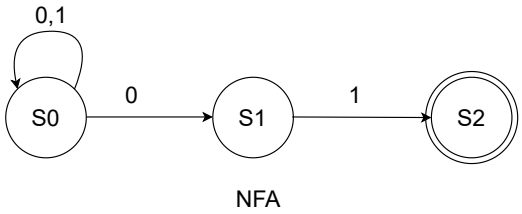


(i) Ex.6

4. Conversia NFA-DFA

Exemplu 1

Să se realizeze conversia NFA-DFA pentru automatul din figura de mai jos:



(j) Ex.1 conversie NFA - DFA

Soluție

Construim un tabel astfel: pornim de la starea inițială și pentru fiecare simbol vom scrie mulțimea stărilor în care se poate ajunge și de fiecare dată când găsim o mulțime nouă, pe care am notat-o mai jos cu roșu, adăugăm o linie nouă în tabel și repetăm operațiile. Fiecare mulțime nouă obținută va deveni o stare nouă în DFA. Tabelul corespunzător automatului de mai sus, scris pas cu pas, este:

starea	0	1
S0	{S0,S1}	S0

(k) Pas 1

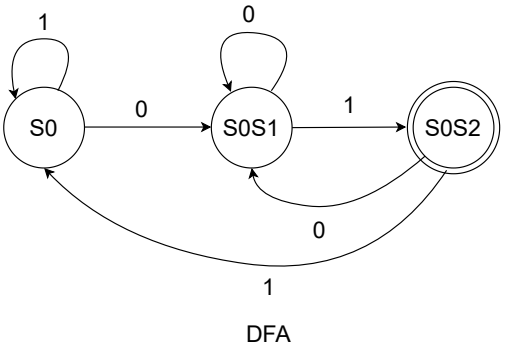
starea	0	1
S0	{S0,S1}	S0
{S0,S1}	{S0,S1}	{S0,S2}

(l) Pas 3

starea	0	1
S0	{S0,S1}	S0
{S0,S1}	{S0,S1}	{S0,S1}
{S0,S2}	{S0,S1}	S0

(m) Pas 3

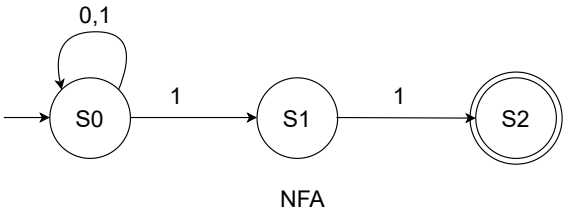
La pasul 3 nu am mai obținut stări noi deci ne vom opri aici. Starea inițială va fi tot S0. Stările acceptoare în DFA-ul obținut sunt stările care conțin o stare acceptoare din automatul inițial (în cazul nostru deoarece S2 era stare acceptoare, atunci și S0S2 va fi stare acceptoare). Pe baza tabelului de tranziție realizăm automatul:



(n) DFA rezultat

Exemplu 2

Să se realizeze conversia NFA-DFA pentru automatul din figura de mai jos:



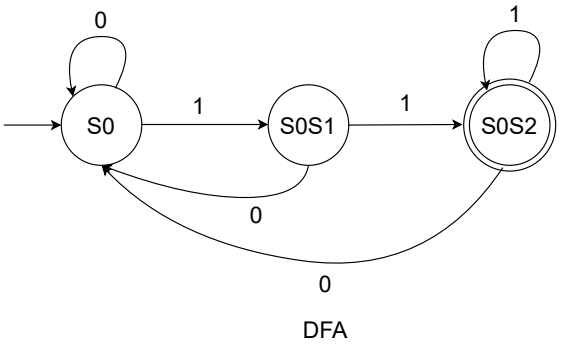
(o) Ex.2 conversie NFA - DFA

Soluție

Construim tabelul în mod asemănător cu cel de la exercițiul precedent:

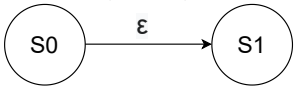
starea	0	1
S0	S0	{S0,S1}
{S0,S1}	S0	{S0,S1,S2}
{S0,S1,S2}	S0	{S0,S1,S2}

Pe baza tabelului de tranziție realizăm automatul:



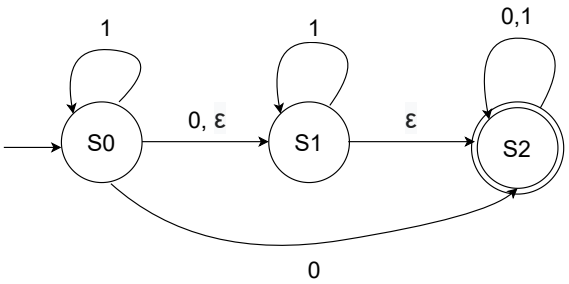
(q) DFA rezultat

O tranziție ε (șirul vid) nu consumă simbol:



Exemplu 3

Să se realizeze conversia NFA-DFA pentru automatul NFA cu tranziții ε din figura de mai jos:



(r) Ex.3 conversie NFA - DFA

Soluție

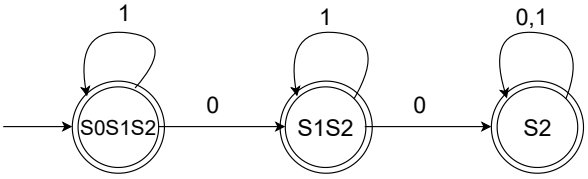
Construim tabelul de tranziții pentru automatul de mai sus, dar ținem cont și de faptul că avem tranziții ε ce se pot realiză fără să consume simbol::

starea	0	1
S0	{S1,S2}	{S0,S1,S2}
S1	S2	{S1,S2}
S2	S2	S2

Pentru cele noi 3 mulțimi de stări apărute (acestea vor fi stările din DFA) realizăm tabelul de tranziție:

starea	0	1
{S0,S1,S2}	{S1,S2}	{S0,S1,S2}
{S1,S2}	S2	{S1,S2}
S2	S2	S2

Deoarece toate cele 3 mulțimi îl conțin pe S2 (care era stare finală), toate cele 3 vor fi stări finale în DFA. Starea inițială va fi S0S1S2.



(s) DFA rezultat

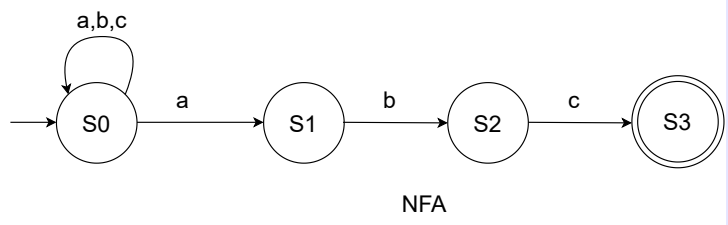
5. Temă

Observație privind încărcarea temei: Pentru tema aceasta încărcați **un singur fișier** (.doc, .docx, .pdf, etc). În cazul în care faceți rezolvarea pe foaie, scanați foile și încărcați pdf-ul (Nu încărcați poze).

Temă

Exercițiul 1:
a) Realizați un automat ce acceptă cuvintele ce a, b sau c și conțin subșirurile ac sau bc, urmate de subșirul bac. Scrieți și o expresie regulată pentru același tip de cuvinte.
b) Ce modificare ați face automatului pentru a accepta orice cuvinte ce conțin a,b sau c, dar să nu conțină subșirurile ac sau bc, urmate de subșirul bac.

Exercițiul 2:
Realizați conversia NFA-DFA pentru:



NFA

Exercițiul 3 (Opțional)
Scrieți o expresie regulată ce poate fi folosită pentru a extrage date despre zborurile dintr-un aeroport. Considerăm că acestea vor fi de forma: Oras1 - Oras2 : DatăPlecare. Numele de Oras începe cu literă mare, apoi poate conține oricâte litere mici. Data va fi de forma DD.MM.YYYY (unde D,M,Y sunt cifre).
Exemplu de șir ce va fi acceptat: ”Timisoara - Barcelona : 25.12.2022”
Un exemplu ce nu va fi acceptat: ”timisoara - BaRceLona : 250.120.20212”
Opțional puteți încerca să îmbunătățiți validarea pentru dată (să nu aveți de exemplu o dată de forma 31.31.30000)