Tema 1 LFA 2015-2016
Limbaje regulate

George Daniel MITRA

17 decembrie 2015

Rezumat

Tema constă în implementarea unui program care afișează toate subșirurile dintr-un text care aparțin unui limbaj dat.

1 Specificații temă

1.1 Cerință

Să se implementeze un program care, primind o reprezentare a unui automat finit determinist și un text, să afișeze toate subșirurile textului care aparțin limbajului automatului.

Soluțiile care nu folosesc FLEX vor fi depunctate după cum e menționat în secțiunea 4.

1.2 Conținutul arhivei

Arhiva trebuie să conțină:

- surse, a căror organizare nu vă e impusă
- un fișier Makefile care să aibă target de build și run (Puteți folosi o versiune modificată a celui din tema 0)
- un fișier README care să conțină numele și grupa și maxim 20 de linii de maxim 80 de caractere în care să menționați algoritmul aplicat pentru construcția automatului și să descrieți abordarea pentru lexer. Cu cât mai scurt, cu atât mai bine!

Absența oricărui element mai sus menționat va duce la nepunctarea temei.

Arhiva trebuie să fie zip. Nu rar, 7z, ace sau alt format ezoteric. Fișierul Makefile trebuie să fie în rădăcina arhivei.

1.3 Format README

Prima linie din README trebuie să conțină numele vostru, așa cum apare pe cs.curs.pub.ro. Prenumele trebuie să fie complet, numele să fie scris cu majuscule, dacă aveți diacritice în nume, ele trebuie să apară și salvați fișierul ca UTF-8. Prima linie ar trebui să arate așa, cu mențiunea că folositi numele vostru:

Nume: MITRA George Daniel

A doua linie trebuie să conțină seria și grupa. Seria și grupa trebuie să fie lipite, fără spații între ele, seria apărând înainte. Exemplu:

Grupa: CB334

Restanțierii de anul patru își vor scrie grupa curentă, nu grupa în care erau când au făcut LFA, punând un R în față. Exemplu:

Grupa: R342C4

Restul textului trebuie să încapă în maxim 20 de linii și fiecare linie să aibă cel mult 80 de caractere.

Scrieți doar ceea ce e esențial!

Fișierul trebuie să se numească README, nu readme, ReadMe, README.txt, readme.txt, read-me.doc, rEADME, README.md sau alte variante asemănătore sau nu. Nerespectarea acestor cerințe duce la nepunctarea temei.

1.4 Specificații program

1.4.1 Intrări

Programul va citi dintr-un fișier numit „input” automatul finit, în formatul specificat mai jos. Programul va citi dintr-un fișier numit „text” un text din care va afișa subșirurile din limbaj.

Se consideră corect orice primește programul ca intrare pentru punctajul maxim. Pentru bonus, programul trebuie să poată raporta erori sintactice și semantice în automat.

Simbolurile din text care nu apar în alfabet se tratează ca separatori. Adică o trecere prin automat care începe înainte de separator se întrerupe și după separator va începe o alta. Acest lucru vă asigură timpi mai mici de rulare fără prea multe optimizări și vă permite să citiți fisierul linie cu linie, fără a fi nevoie să le concatenați.

1.4.2 Ieșiri

Programul va afișa la ieșirea standard rezultatul, câte un subșir pe linie. Șirul vid nu se va afișa fiindcă nu are sens să căutăm șirul vid într-un text.

1.4.3 Erori

În cazul în care automatul nu respectă sintaxa specificată în secțiunile 2.2 și 3.2, programul trebuie să afișeze mesajul „Syntax error”. Dacă fișierul lex e bine făcut, cazurile de erori sintactice se rezolvă cu o singură regulă la final.

În cazul în care automatul nu are erori sintactice, dar folosește undeva stări care nu sunt în mulțimea stărilor, K, simboluri care nu sunt în alfabet, Σ, sau are numărul de tranziții pentru o combinatie (stare sursă, simbol) diferit de 1, programul trebuie să afișeze mesajul „Semantic error”. Erorile semantice pot apărea la nivel de tranziții(tranziții lipsă, tranziții duplicat, stări incorecte, simboluri incorecte), stare inițială sau stări finale.

Erorile se afișează la ieșirea standard; e mai simplu așa.

2 Noțiuni introductive

2.1 Limbajul de descriere

Limbajul este descris printr-o gramatică BNF și folosește aceeași convenție de culori ca articolul Wikipedia despre BNF:

- albastru - neterminali
- verde - operatori ai limbajului BNF și paranteze ajutătoare
- rosu - terminali (elemente care fac parte efectiv din limbajul descris)

2.2 Simbol, Alfabet, Șir

2.2.1 Simbol

Atenție, se folosesc simboluri diferite față de tema 0:

<symbol> ::= <upper-case letter> | <lower-case letter> | <digit> | <other>

<lower-case letter> ::= (a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z)

<upper-case letter> ::= (A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z)

<digit> ::= (0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9)

<other> ::= (! | # | $ | % | & | - | . | / | : | ; | < | > | = | @ | [|] | ? | + | ^ | ‘ | ´ | ¨ | _ | * | ~ | |)

2.2.2 Alfabet

<alphabet> ::= { <symbol> (, <symbol>)* }

2.2.3 Șir

Atenție, un cuvânt e definit diferit față de cazul expresiilor regulate, nemaiavând șirul vid:

<word> ::= (<symbol>)+

3 Automate Finite Deterministe(AFD)

3.1 Descriere

Un automat finit este un model de calcul care primește la intrare o bandă de simboluri și își modifică starea internă în funcție de ce are la intrare. Acesta poate fi privit ca o cutie neagră cu niște stări între care face tranziții în funcție de intrare, la fiecare tranziție mișcând capul de citire la dreapta.

Formal, un automat finit determinist este un tuplu $M = (K, \Sigma, \delta, s, F)$, cu următoarele proprietăți:

- K este mulțimea stărilor. K este finită, de unde vine și numele automatului;
- Σ este alfabetul din care sunt formate cuvintele acceptate de automat;

- δ se numește funcție de tranziție. $\delta : K \times \Sigma \to K$. $\delta(p, a) = q; p, q \in K; a \in \Sigma$ înseamnă că automatul, dacă se află în starea p și primește pe bandă a, trece în starea q. Fiindcă δ este funcție, toate tranzițiile din fiecare stare pentru fiecare simbol trebuie să fie definite.
- $s \in K$ este starea de start. Aceasta este starea în care se află automatul înainte de a primi ceva pe bandă.
- $F \subseteq K$ este mulțimea stărilor finale. Dacă automatul se află într-o stare $f \in F$ după ce a terminat de citit șirul înseamnă că acceptă șirul.

3.2 Specificații

În temă, un automat finit determinist este dat ca un tuplu, conform definiției:
<DFA> ::= (<states> , <alphabet> , <transitions> , <state> , (<states> | {}))
<states> ::= { <state> (, <state>)* }
<state> ::= <name>
<name> ::= (<upper-case letter> | <lower-case letter> | <digit> | _) +
<transitions> ::= (<transition> (, <transition>) *)
<transition> ::= d(<state> , <symbol>)= <state>

4 Punctaj

4.1 Checker

Checker-ul va oferi un punctaj între 0 și 150. Ce este peste 100 intră în categoria bonus. Testele sunt publice.

4.2 Bonus și Depunctări

Implementările care nu folosesc flex primesc maxim 50% din punctaj.

Pentru temele care nu respectă specificațiile la rularea automată, punctajul va fi 0.

Deadline: 20.01.2016, 23:59. Upload-ul va rămâne deschis până la ora 05:00.

5 Sugestii

Vă recomand să începeti cât mai repede. E mai ușor decât credeti.

Bibliografie

[1] flex homepage

[2] Lexical Analysis with Flex

[3] Using flex

[4] jflex homepage

[5] jflex user manual

[6] jflex user manual in japanese

[7] Laborator 1 SO: Makefile