(Categ1) Construirea unui automat finit pentru o gramatică regulată

Enunţ

Să se scrie o aplicaţie care pentru o gramatică generativă regulată $G = (V_N, V_T, S, P)$ construieşte un automat finit $M = (Q, \Sigma, \delta, q_0, F)$, pentru care $T(M) = L(G)$.

Cerinţe

I. Se cere definirea unei clase **Grammar** (alta decât clasa principală - aviz celor care programează Java sau C#!).

Membrii clasei vor fi: V_N, V_T, S, P - pentru mulţimea de producţii se poate utiliza o clasă/structură.

Metodele clasei - obligatorii. (Pot exista şi altele, dacă este necesar)

1. **VerifyGrammar** - verifică dacă gramatica citită este o gramatică corectă/validă. Aici trebuie să identificaţi din teorie, care sunt verificările necesare.
2. **IsRegular** - verifică dacă gramatica este regulată sau nu.
3. **GenerateWord()** - generează un cuvânt pornind de la simbolul de start. La generarea unui cuvânt în gramatică, producţia care se aplică la momentul curent trebuie aleasă random dintre producţiile aplicabile la momentul curent. Se vor afişa toate combinaţiile prin care s-a trecut până la generarea cuvântului.
4. **PrintGrammar()** - afişarea frumoasă a elementelor gramaticii. Aici se poate lua în calcul şi supraîncărcarea operatorului specific.
5. **ReadGrammar()** - citirea tuturor elementelor gramaticii. Aici se poate lua în calcul şi supraîncărcarea operatorului specific.

II. Se cere definirea unei clase **FiniteAutomaton**

Membrii clasei vor fi: $(Q, \Sigma, \delta, q_0, F)$ (daţi denumiri semnificative acestor membri).

Observaţie: definirea trebuie să fie suficient de generală, încât să cuprindă şi AFN şi AFD.

Metodele clasei - obligatorii. (Pot exista şi altele, dacă este necesar)

1. **VerifyAutomaton** - verifică dacă este un automat valid. Aici trebuie să identificaţi din teorie, care sunt verificările necesare.
2. **PrintAutomaton** - afişarea frumoasă a elementelor unui automat. Aici se poate lua în calcul şi supraîncărcarea operatorului specific.
3. **CheckWord** - o funcţie care verifică dacă un cuvânt este acceptat sau nu de un automat.
4. **IsDeterministic** - verifică dacă automatul este determinist sau nu.

III. Definiţi o funcţie care preia o gramatică regulată G şi returnează un obiect de tip automat. Automatul returnat trebuie să recunoască limbajul generat de G. Funcţia poate fi membră a clasei **Grammar** sau nu.

IV. În funcţia principală:

1. Se citesc din fişier elementele unei gramatici regulate. Se verifică dacă gramatica este corectă şi regulată. Numai în caz afirmativ meniul devine disponibil.
2. Se crează un meniu (care se afişază până când nu mai sunt dorite opţiuni) care permite:
   (a) afişarea gramaticii G
   (b) generarea unui număr n de cuvinte în gramatica G
   (c) obţinerea automatului echivalent cu G. Automatul echivalent se afişează.
   (d) verificarea dacă un cuvânt este sau nu acceptat de în automatul obţinut .
   (e) generarea unui cuvânt în G + verificarea dacă e acceptat de către automat

V. BAREM:

1. Definire corectă a clasei Grammar, cu membrii corespunzători - 1pct
2. Metoda I. 1 - 0.5pct
3. Metoda I. 2 - 0.5pct

4. Metoda I. 3 - 1 pct, dacă este complet rezolvată cf cerinţei
5. Metode I. 4 & I. 5 - 0.5 pct
6. Definire corectă a clasei FiniteAutomaton, cu membrii corespunzători - 1pct
7. Metoda II. 1 - 0.5pct
8. Metoda II. 2 - 0.25pct
9. Metoda II. 3 - 1pct
10. Metoda II. 4 - 0.5pct
11. Algoritm III - 1.75pct
12. Etapa IV - 0.25pct pentru manipularea corectă a fişierului + 0.25pct pentru meniu