

(Categ1) Construirea unui automat finit pentru o gramatică regulată

Enunț

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

Cerințe

- 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 mulțimea de producții se poate utiliza o clasă/structură.

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

1. **VerifyGrammar** - verifică dacă gramatica citită este o gramatică corectă/validă. Aici trebuie să identificați 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, producția care se aplică la momentul curent trebuie aleasă random dintre producțiile aplicabile la momentul curent. Se vor afișa toate combinațiile prin care s-a trecut până la generarea cuvântului.
4. **PrintGrammar()** - afișarea frumoasă a elementelor gramaticii. Aici se poate lua în calcul și supraîncărcarea operatorului specific.
5. **ReadGrammar()** - citirea tuturor elementelor gramaticii. Aici se poate lua în calcul și supraîncărcarea operatorului specific.

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

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

Observație: definirea trebuie să fie suficient de generală, încât să cuprindă și AFN și AFD.

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

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

III. Definiți o funcție care preia o gramatică regulată G și returnează un obiect de tip automat. Automatul returnat trebuie să recunoască limbajul generat de G . Funcția poate fi membră a clasei **Grammar** sau nu.

IV. În funcția principală:

1. Se citesc din fișier elementele unei gramatici regulate. Se verifică dacă gramatica este corectă și regulată. Numai în caz afirmativ meniul devine disponibil.
2. Se crează un meniu (care se afișază până când nu mai sunt dorite opțiuni) care permite:
 - (a) afișarea gramaticii G
 - (b) generarea unui număr n de cuvinte în gramatica G
 - (c) obținerea automatului echivalent cu G . Automatul echivalent se afișează.
 - (d) verificarea dacă un cuvânt este sau nu acceptat de în automatul obținut .
 - (e) generarea unui cuvânt în G + verificarea dacă e acceptat de către automat

V. BAREM:

1. Definiere corectă a clasei Grammar, cu membrii corespunzători - 0.25pct
2. Metoda I. 1 - 0.5pct
3. Metoda I. 2 - 0.5pct

4. Metoda I. 3 - 2 pct, dacă este complet rezolvată cf cerinței
5. Metode I. 4 & I. 5 - 0.25 pct
6. Definire corectă a clasei FiniteAutomaton, cu membrii corespunzători - 0.25pct
7. Metoda II. 1 - 0.5pct
8. Metoda II. 2 - 0.25pct
9. Metoda II. 3 - 1.5pct
10. Metoda II. 4 - 0.5pct
11. Algoritm III - 2pct
12. Etapa IV - 0.5pct.