Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică și Microelectronică

Programul de studiu Managementul Informației

**Lucrarea de laborator nr. 1**

la disciplina

***Limbaje formale și compilatoare***

Elaborat:

stud. gr. Calancea Catalin MI-222

Verificat: asis. univ. dr. Gaidarji Alina

Chișinău, 2024

**Lucrarea de laborator N1 la disciplina LFPC**

**Sarcina lucrării:**

1. Pentru gramatica formală **G = (VN, VT, P, S)** construiţi 5 şiruri, care aparţin limbajului L(G) generat de această gramatică. Lungimea şirului trebuie să fie nu mai mică, decât numărul de caractere din alfabet VN+2.
2. Pentru fiecare şir să se construiască arborii de derivare.
3. Desenaţi automatul finit echivalent acestei gramatici.
4. La ce clasă al gramaticilor după Chomsky aparţine gramatica dată?

VN={S, L,D },

VT={a, b,c,d,e,f,j} ,

P= {

1. S → aS
2. S → bS
3. S → cD
4. S → dL
5. S → e
6. L → eL
7. L → fL
8. L → jD
9. L → e
10. D → eD
11. D → d}

**Efectuarea lucrării:**

* 1. Construim 5 şiruri ce aparţin limbajului L(G) generat de această gramatică:

1. S->aS->adL->adeL->adee
2. S->aS->abS->abcD->abcd
3. S->dL->dfL->dfiD->dfid
4. S->e

S->dL->dfL->dfeL->dfee

* 1. Pentru fiecare şir construim arborii de derivare:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S  / |  a S  / |  d L  / |  e L  |  e | S  / \  a S  / \  b S  / \  c D  |  d | S  /\  d L  /\  f L  /\  j D  |  d | S  |  e | S  /\  d L  /\  f L  /\  e L  |  e |

* 1. Construim AF(automatul finit) echivalent acestei gramatici:

**AF=(Q, Σ, δ, q0, F), unde**

**Q – mulţimea de stări**

**Σ - vocabular**

**δ - funcţia de tranziţie**

**q0 – starea iniţială**

**F – mulţimea stărilor finale**

**Algoritmul de construire AF:**

1. Q = VN∪{X}={S, L, D, X}
2. Σ=VT={ a, b, c, d, e, f, j}
3. q0=S
4. F={X}
5. Pentru toate producţiile definim **δ**:

Iniţial toate mulţimile δ(A, b): = 0

1. S →aS

δ(S, a): = δ (S, a)∪{S}={S}

1. S →bS

δ(S, b): = δ (S, b)∪{S}={S}

1. S →cD

δ(S, c): = δ (S, c)∪{D}={D}

1. S →dL

δ(S, d): = δ (S, d)∪{L}={L}

1. S →e

δ(S, e): = δ (S, e)∪{X}={S, X}

1. L →eL

δ(L, e): = δ (L, e)∪{L}={L}

1. L →fL

δ(L, f): = δ (L, f)∪{L}={L}

1. L → jD

δ(L, j): = δ (L, j)∪{D}={D}

1. L →eL

δ(L, e): = δ (L, e)∪{L}={L}

1. D → eD

δ(D, e): = δ (D, e)∪{D}={D

1. D →d

δ(D, d): = δ (D, d)∪{X}={D, X}

Codul in C:

#include <stdio.h>

#include <string.h>

void regula1(char\* arr);

void regula2(char\* arr);

void regula3(char\* arr);

void regula4(char\* arr);

void regula5(char\* arr);

int main() {

char arr[10];

char vN[1];

printf("Cuvinte dupa Regula 1:\n");

regula1(arr);

printf("\nCuvinte dupa Regula 2:\n");

regula2(arr);

printf("\nCuvinte dupa Regula 3:\n");

regula3(arr);

printf("\nCuvinte dupa Regula 4:\n");

regula4(arr);

printf("\nCuvinte dupa Regula 5:\n");

regula5(arr);

printf("\n\nAutomatul finit echivalent gramaticii date este:");

printf("\nAF=(Q, SI, d, q0, F)\n");

printf("Q = VN U {X} = {S, L, D, X}\n");

printf("SI = VT = {a,b, c, d, e f, j}\n");

printf("q0 = S\n");

printf("F = {X}\n");

if (vN[1]) {

printf("\n1. S -> aS \nd(S,a) := d(S,a) U {S}={S}\n\n");

printf("2. S -> bS \nd(S,b) := d(S,b) U {S}={S}\n\n");

printf("3. S -> cD \nd(S,c) := d(S,c) U {D} = {D}\n\n");

printf("4. S -> dL \nd(S,d) := d(S,d) U {L} = {L}\n\n");

printf("5. S -> e \nd(S,e) := d(S,e) U {X} = {S,X}\n\n");

printf("6. L -> eL \nd(L,e) := d(L,e) U {L} = {L}\n\n");

printf("7. L -> fL \nd(L,f) := d(L,f) U {L} = {L}\n\n");

printf("8. L -> jD \nd(L,j) := d(L,j) U {D} = {D}\n\n");

printf("9. L -> eL \nd(L,e) := d(L,e) U {L} = {L}\n\n");

printf("10. D -> eD \nd(D,e) := d(D,e) U {D} = {D}\n\n");

printf("11. D -> d \nd(D,d) := d(D,d) U {X} = {D,X}\n\n");

}

return 0;

}

void regula1(char\* arr) {

arrcpy(arr, "a");

printf("%s -> ", arr);

arrcat(arr, "d");

printf("%s -> ", arr);

arrcat(arr, "e");

printf("%s -> ", arr);

arrcat(arr, "e");

printf("%s ", arr);

}

void regula2(char\* arr) {

arrcpy(arr, "a");

printf("%s -> ", arr);

arrcat(arr, "b");

printf("%s -> ", arr);

arrcat(arr, "c");

printf("%s -> ", arr);

arrcat(arr, "d");

printf("%s ", arr);

}

void regula3(char\* arr) {

arrcpy(arr, "d");

printf("%s -> ", arr);

arrcat(arr, "f");

printf("%s -> ", arr);

arrcat(arr, "j");

printf("%s -> ", arr);

arrcat(arr, "d");

printf("%s ", arr);

}

void regula4(char\* arr) {

arrcpy(arr, "e");

printf("%s ", arr);

}

void regula5(char\* arr) {

arrcpy(arr, "d");

printf("%s -> ", arr);

arrcat(arr, "f");

printf("%s -> ", arr);

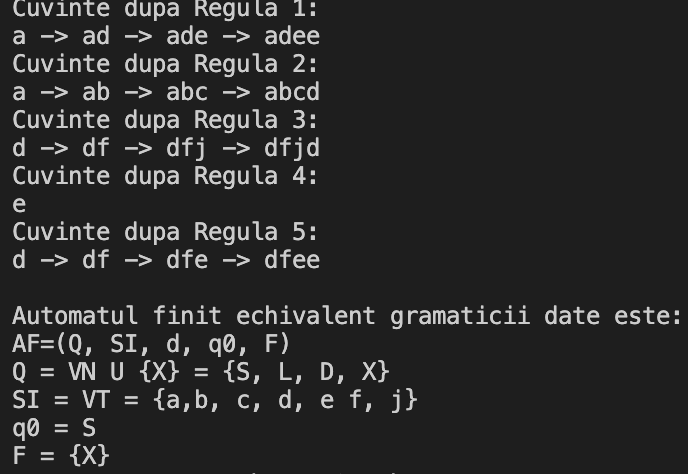
arrcat(arr, "e");

printf("%s -> ", arr);

arrcat(arr, "e");

printf("%s ", arr);

}



**CONCLUZIE**

În concluzie, gramaticile regulate sunt un tip de gramatică formală care poate fi utilizată pentru a defini și recunoaște limbaje regulate. Acestea sunt alcătuite dintr-un set finit de reguli de producție, care permit doar transformări simple asupra simbolurilor terminale și neterminale.

Un exemplu de gramatică regulată este gramatica regulată a expresiilor regulate, care este utilizată în teoria automatelor și limbajelor formale. Această gramatică poate fi folosită pentru a defini și genera expresii regulate, care sunt apoi utilizate pentru a recunoaște șiruri de caractere.

În plus, gramaticile regulate sunt importante în informatică, în special în compilatoarele de limbaje de programare, deoarece multe dintre acestea sunt bazate pe limbaje regulate. În general, învățarea despre gramatici regulate este un aspect important al studiului teoriei limbajelor formale și al aplicațiilor practice ale acesteia.