Gramatici generative

Lab intermediar, fără notă finală

Să se implementeze o **gramatică generativă** astfel:

* Se citesc din fișier elementele componente: VN , VT, simbolul de start și producțiile;
* Se citește de la tastatură un număr n, iar apoi se generează n cuvinte *distincte* în gramatică. Generarea are loc în modul următor: se pornește de fiecare dată de la simbolul de start, iar cât timp există producții aplicabile, se aplică *random* una dintre acestea;

Se cere creearea unei clase **Gramatica**.

Membrii clasei vor fi: **VN**, **VT**, **S**, **producțiile (P).**

În plus, clasa are metodele:

* **citire** - citirea elementelor gramaticii
* **verificare**:

(1) VN ∩VT = ∅

(2) S ∈ VN

(3) pentru fiecare regulă, membrul stâng conține cel puțin un neterminal

(4) există cel puțin o producție care are în stânga doar S

(5) fiecare producție conține doar elemente din VN și VT

* **afișare** - afișarea frumoasă a elementelor gramaticii
* **generare** - generarea de cuvinte pornind de la simbolul de start, cu afișarea fiecărui

pas

# Exemplu:

Fie gramatica: G = ({S,A}, {a,b}, S, P), P conținând următoarele producții:

(1) S -> abS

(2) S -> aSAb

(3) S -> aA

(4) A -> aAb

(5) A -> a

Etape:

- Pornim generearea unui cuvânt de la simbolul de start S

- În acest moment sunt aplicabile producțiile (1), (2) și (3). Se alege random una dintre ele și se aplică. Spre exemplu, dacă funcția random o alege pe a doua, se obține:

S => aSAb.

- În acest moment sunt aplicabile toate producțiile. Presupunem că funcția random o va alege din nou pe a doua:

S => aSAb => aaSAbAb

- Din nou sunt aplicabile toate producțiile. Să presupunem că funcția random va alege producția (4). În acest moment producția este aplicabilă în două locuri. Locul în care se va aplica producția se va alege din nou random. Spre exemplu, se poate alege înlocuirea primului caracter A.

S => aSAb => aaSAbAb => aaSaAbbAb

- Se continuă în același mod până se obține un cuvânt alcătuit doar din terminale.

Exemplu:

S => aSAb => aaSAbAb => aaSaAbbAb => aaaaAaAbbAb => aaaaaaAbbAb => aaaaaaAbbab => aaaaaaabbab

# Sugestii de implementare:

* Pentru a genera o clasă Gramatică, este nevoie de membrii menționați mai sus, atenție la producții. Producțiile sunt de forma membruStang -> membruDrept.
* Cum se pot formaliza **producțiile**?

1. O altă clasă definită cu 2 stringuri (membruStang și membruDrept), iar apoi Gramatica conține un vector de astfel de producții (asociere între clase).
2. Producțiile reținute ca perechi de componente de tip șiruri de caractere.

Ex (C++) pair<string, string>

(Java) pair<String, String>

(C#) Tuple<string, string>

(Python) Tuple[str, str]

* Din toată lista/vectorul de reguli posibile se alege mereu random o regulă din cele fesabile/aplicabile

(C++) Vezi diverse variante de accesare random <https://www.techiedelight.com/get-random-value-stl-containers-cpp/>

(C++)

#include <iostream>

#include <random>

#include <vector>

int main()

{

const std::vector<int> intList{ 1, 3, 5, 2 };

std::random\_device rd;

std::mt19937 eng(rd());

std::uniform\_int\_distribution<> distr(0, intList.size() - 1);

std::cout<<intList[distr(eng)]<<" ";

}

(Java)

Random class + Random.nextInt(bound)

(C#)

Random class + Random.next(bound)

(Python)

random module + random.choice(list)

* Atenție că o dată ce s-a ales o regulă aplicabilă (random), se înlocuiește o poziție tot random din cuvântul procesat
* Cuvintele generate trebuie să se introducă într-un container care nu permite duplicate.
* Pentru afișare, o producție poate fi afișată sub forma S -> abS
* Pentru *lambda* se poate utiliza un caracter rezervat