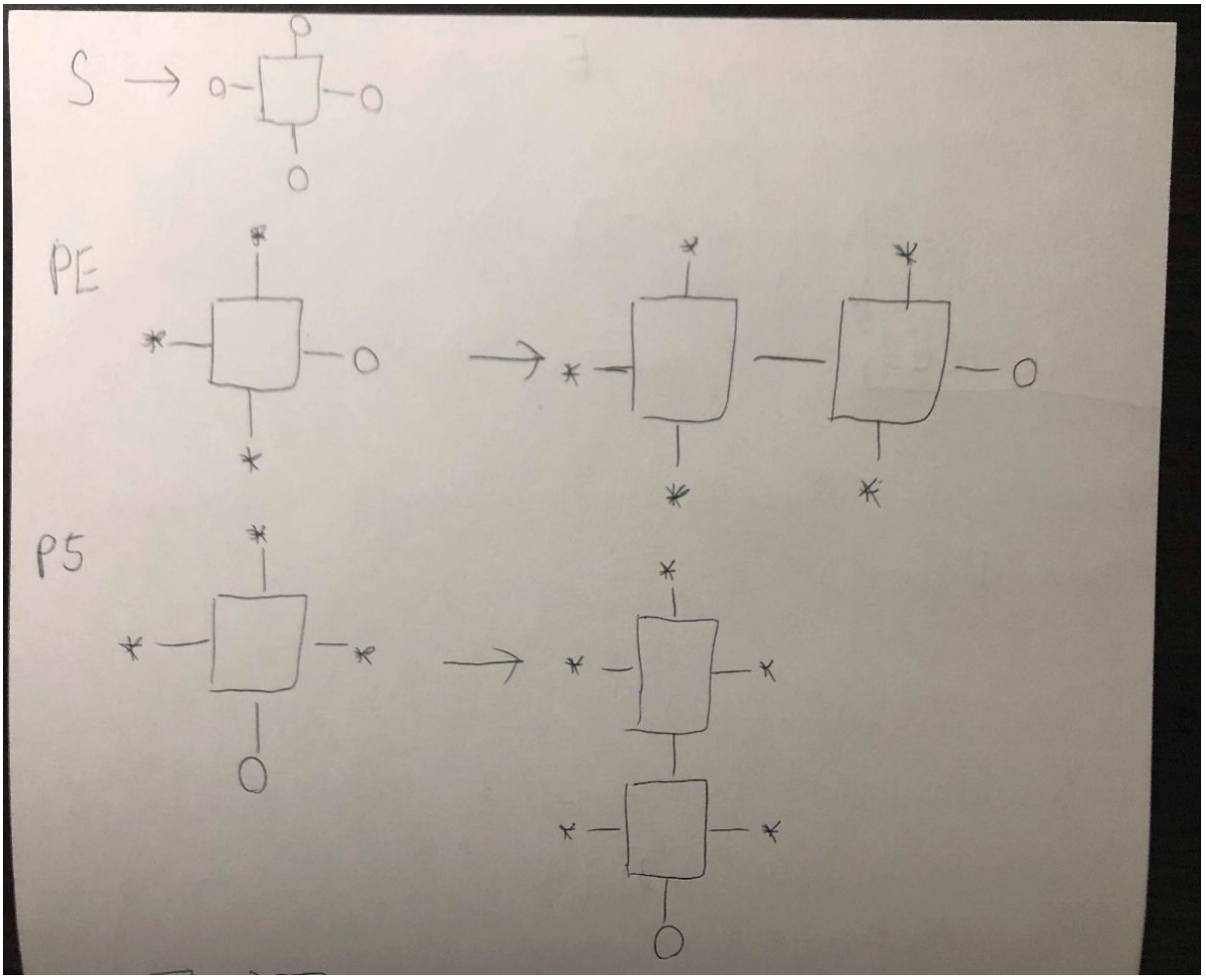
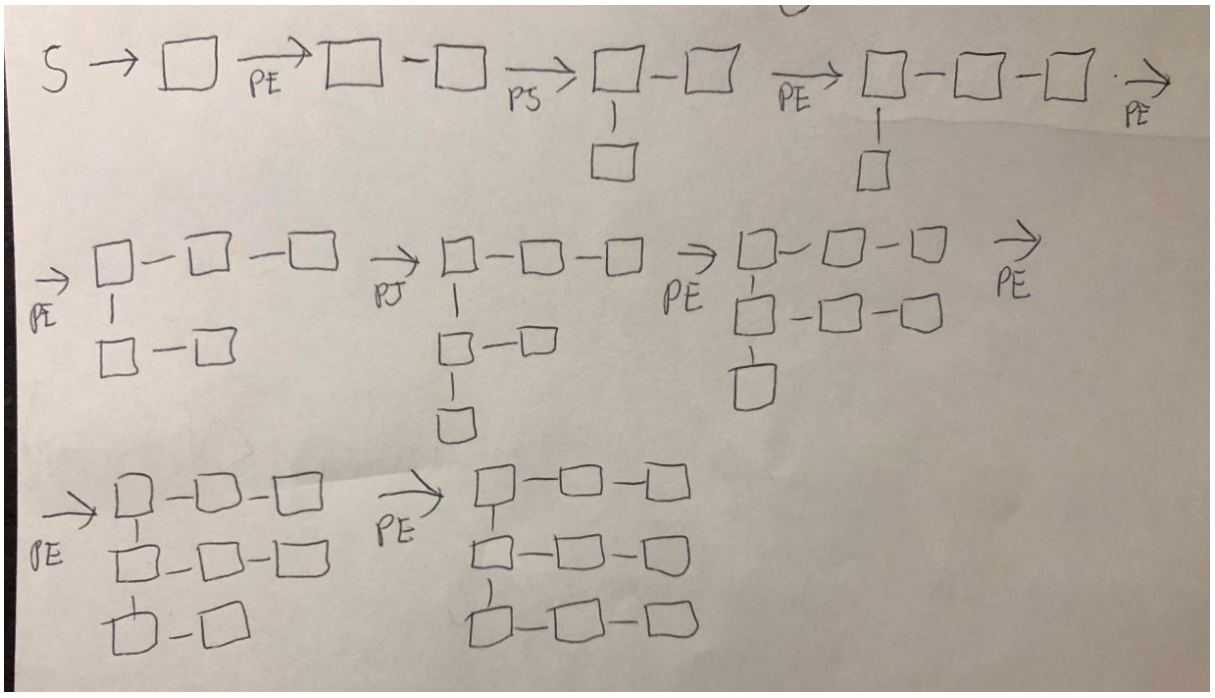
Sprawozdanie z zadania domowego z laboratorium 3. - Dawid Witek 14:40

• Proszę rozszerzyć gramatykę w taki sposób, aby była możliwa generacja siatek prostokątnych, dwuwymiarowych, o ilości elementów N × M

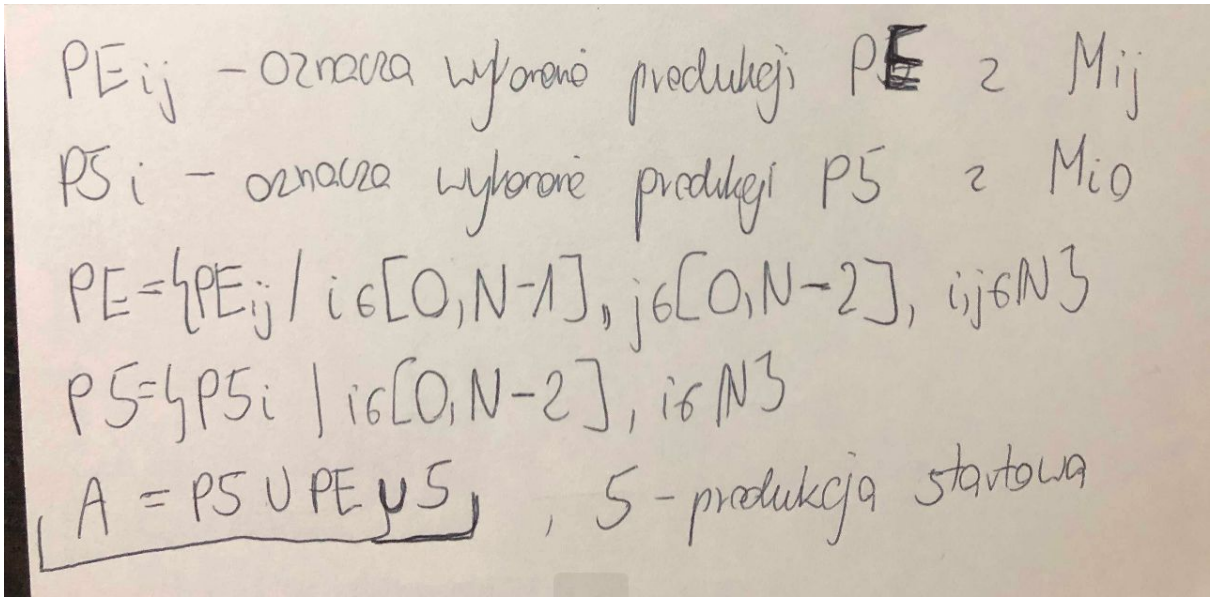


• Proszę napisać ciąg produkcji w gramatyce generujący siatkę prostokątną o 3 × 3 elementach

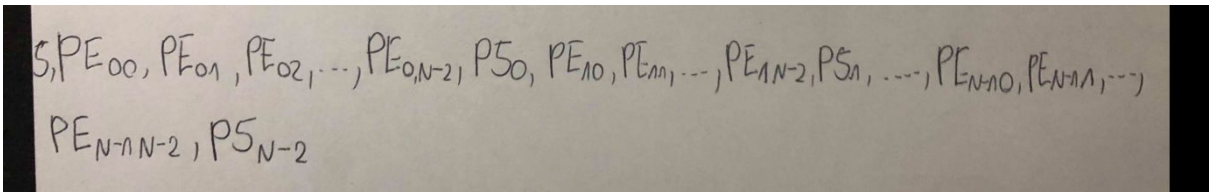


• Bazując na ciągu produkcji w gramatyce generującej przedstawioną siatkę, proszę wskazać alfabet w sensie teorii śladów

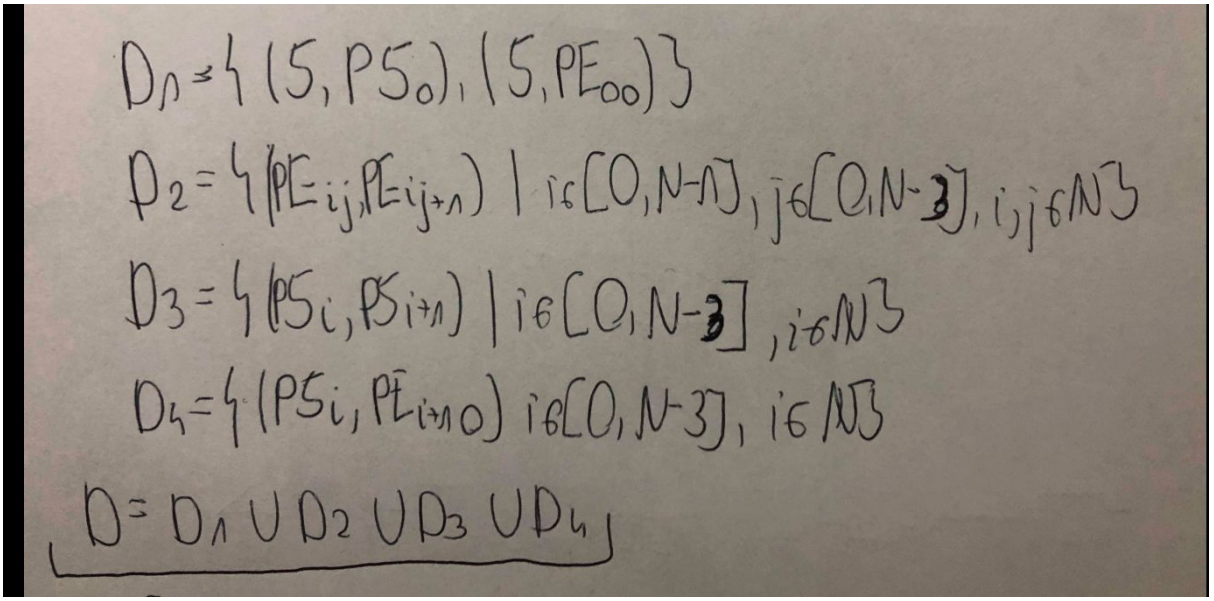
Uogólniłem alfabet dla dowolnej macierzy M o rozmiarze NxN



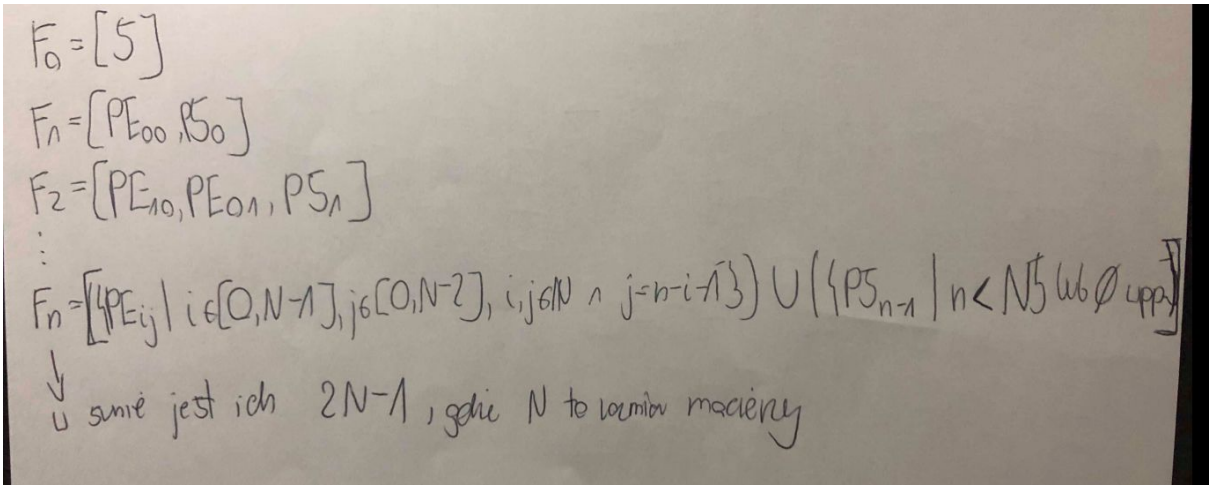
• Proszę napisać słowo (ciąg symboli z alfabetu) odpowiadających generacji siatki prostokątnej



• Proszę wskazać relacje (nie)zależności dla alfabetu, w sensie teorii śladów



• Proszę przekształcić ciąg symboli (słowo) do postaci normalnej Foaty



• Proszę zaprojektować i zaimplementować algorytm współbieżny w oparciu o postać normalną Foaty. Parametr algorytmu to N = ilość kwadratów na każdym boku siatki

Kluczowe zmiany w schedulerze z ćwiczeń:

- wymiana produkcji na odpowiednie dla tego problemu

```
public class PE extends AbstractProduction<Vertex> {
   private Vertex[][] array;

   public PE(Vertex _obj, PDrawer<Vertex> _drawer, Vertex[][]
array){
       super(_obj, _drawer);
       this.array = array;
   }

   @Override
   public Vertex apply(Vertex oldPoint){
       Vertex newPoint = new Vertex(oldPoint.x+1, oldPoint.y);
       array[newPoint.y][newPoint.x] = newPoint;
       return newPoint;
   }
}
```

```
public class PS extends AbstractProduction<Vertex> {
   private Vertex[][] array;

   public PS(Vertex _obj, PDrawer<Vertex> _drawer, Vertex[][]
array){
       super(_obj, _drawer);
       this.array = array;
   }

   @Override
   public Vertex apply(Vertex oldPoint){
       Vertex newPoint = new Vertex(oldPoint.x, oldPoint.y+1);
       array[newPoint.y][newPoint.x] = newPoint;
       return newPoint;
   }
}
```

- implementacja własnego Executora, a dokładniej metody run()

```
public void run() {

   PDrawer drawer = new GraphDrawer();
   //axiom
   Vertex s = new Vertex(0,0);
   int N = 246;
   Vertex[][] array = new Vertex[N][N];
   array[0][0] = s;

   for(int n=1; n<=2*N-2; n++){
       for(int i=0; i<N; i++){
           int j = n-i-1;
           if(j>=0 && j<N-1){
               Vertex oldPoint = array[i][j];
               PE pe = new PE(oldPoint, drawer, array);
               this.runner.addThread(pe);
           }
       }
       if(n<N){
           Vertex oldPoint = array[n-1][0];
           PS ps = new PS(oldPoint, drawer, array);
           this.runner.addThread(ps);
       }
       this.runner.startAll();
   }

   for(int i=0; i<N; i++){
       for(int j=0; j<N; j++){
           if(array[i][j] == null){
               System.out.println("Something went wrong, x:" + j + ",
y:" + i);
           }
       }
   }
   System.out.println("All good :)");
}
```

Całe archiwum z kodem schedulera dołączyłem do sprawozdania.