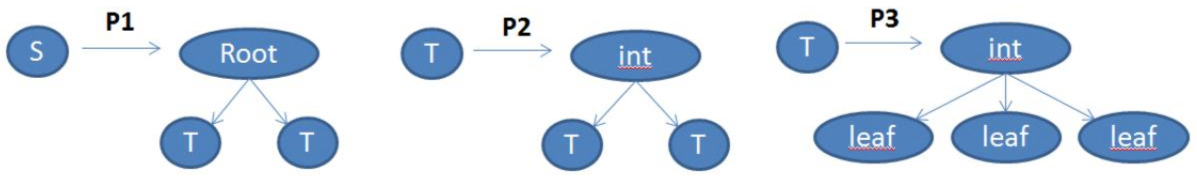
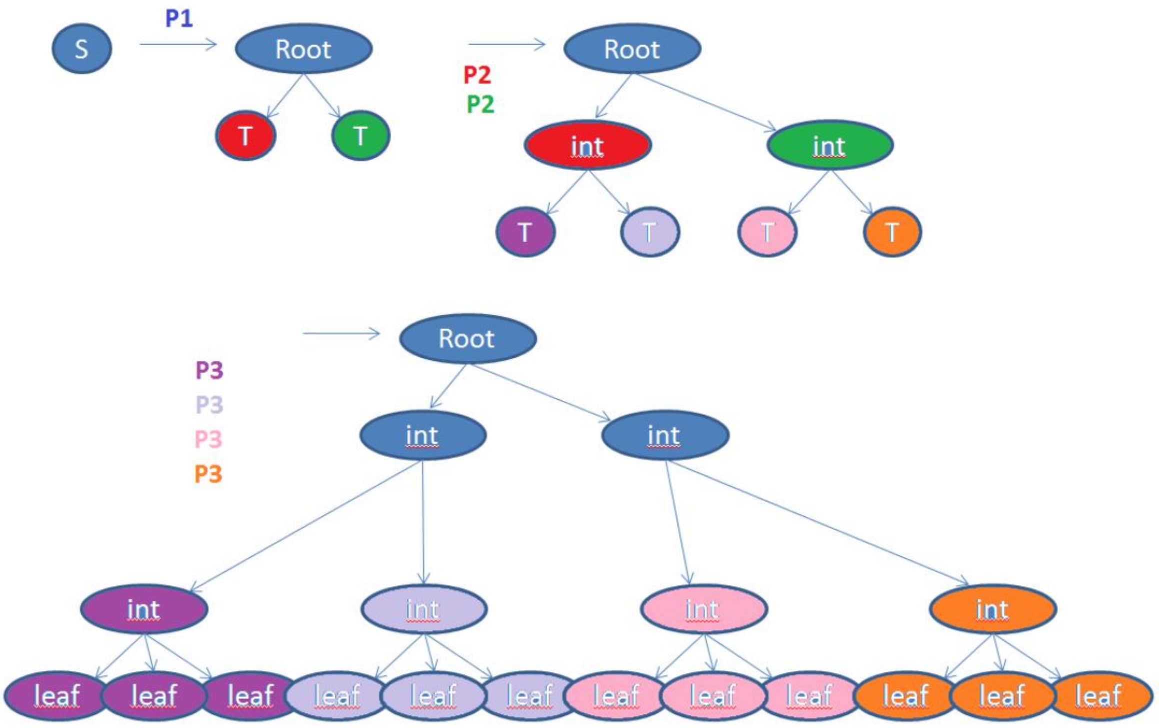
In the exemplary JAVA code we have an Executor class where the tree grammar for scheduled to construct binary tree using the following productions



The derivation of the tree is managed by the following sequence of productions

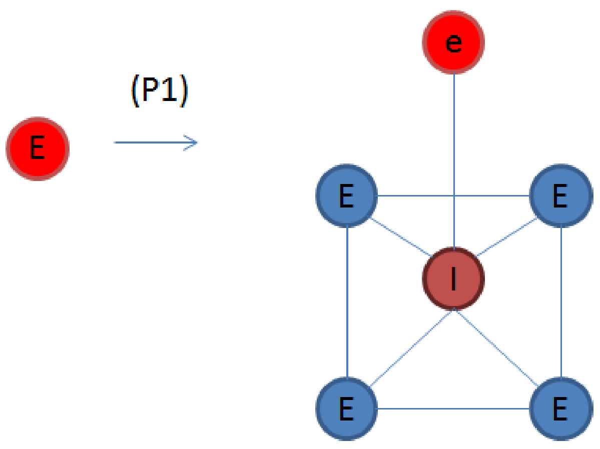


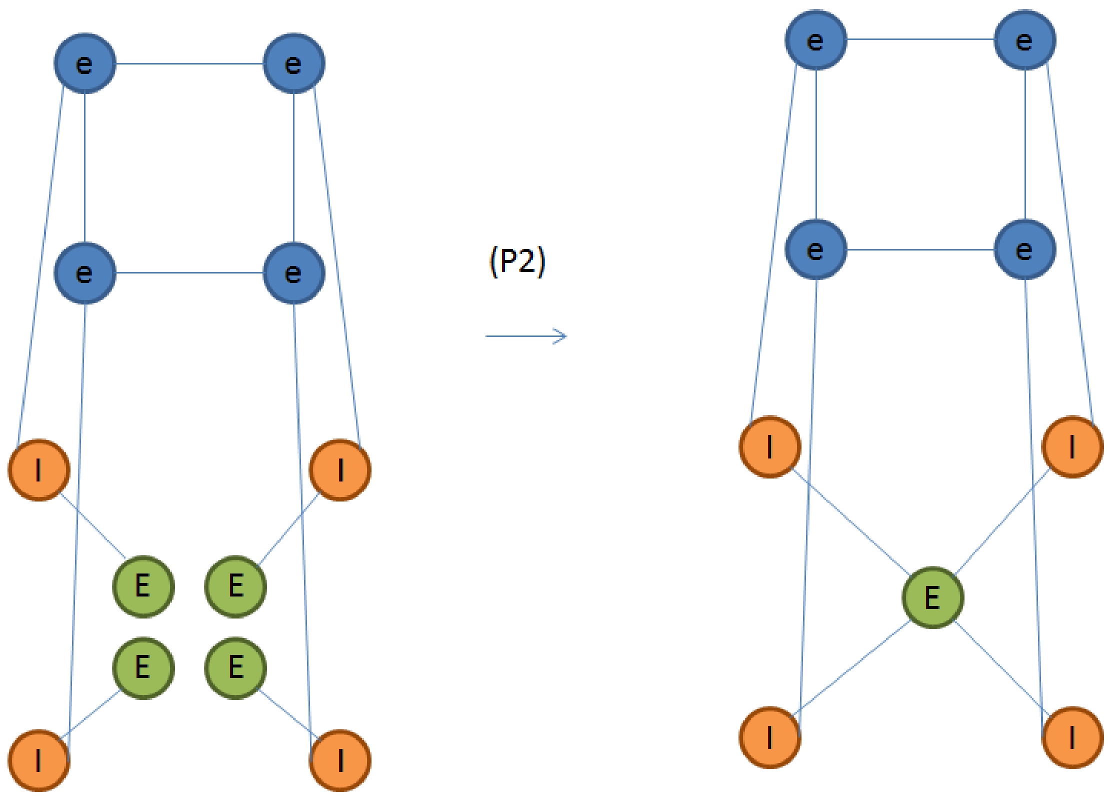
The concurrent scheduling of the productions is managed in the Executor class

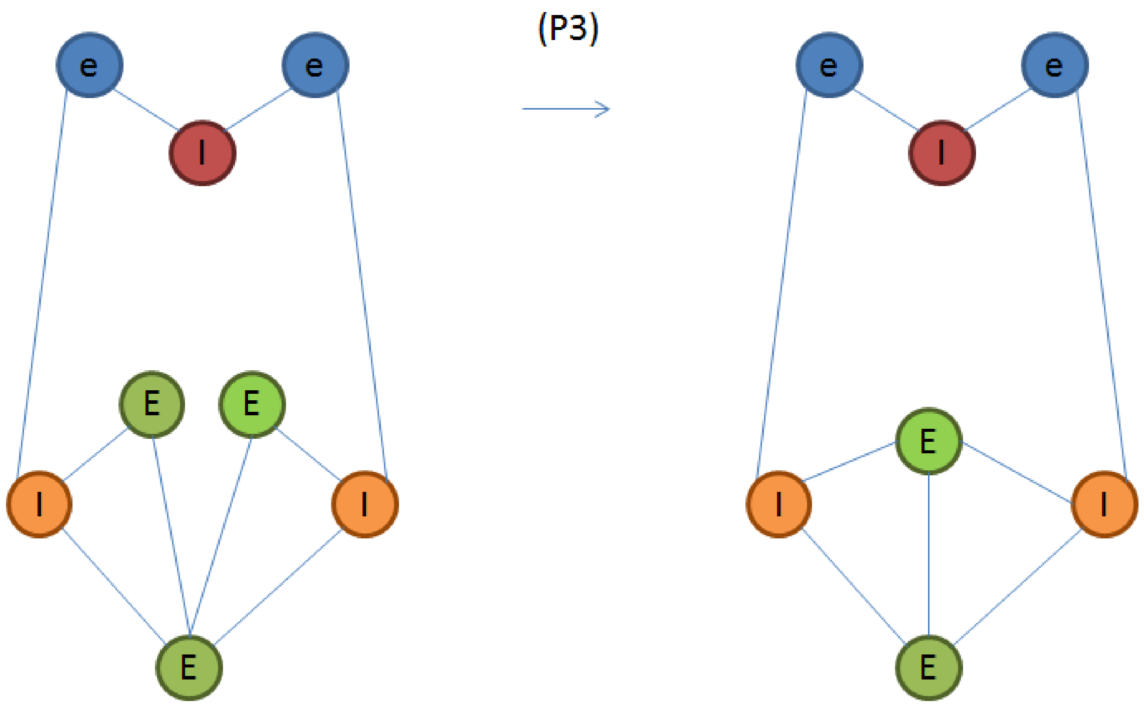
```
class Executor extends Thread {
public synchronized void run() {

// BUILDING TREE

try {
double dx=12.0; //mesh size along x
double dy=12.0; //mesh size along y
int n=12; //number of elements along x axis
int m=12; //number of elements along y axis
int p=2; //polynomial order of approximation

// Mesh
MeshData mesh = new MeshData(dx,dy,n,m,2);
Vertex S = new Vertex(null,null,null,null,"S",0,mesh.m_nelemx,mesh);

//[(P1)]
CyclicBarrier barrier = new CyclicBarrier(1+1);
P1 p1 = new P1(S,barrier, mesh);
p1.start();
barrier.await();
//[(P2)1(P2)2]
barrier = new CyclicBarrier(2+1);
P2 p2a = new P2(p1.m_vertex.m_left,barrier, mesh);
P2 p2b = new P2(p1.m_vertex.m_right,barrier, mesh);
p2a.start();
p2b.start();
barrier.await();

//[(P3)1(P3)2(P3)3(P3)4]
barrier = new CyclicBarrier(4+1);
P3 p3a1 = new P3(p2a.m_vertex.m_left,barrier, mesh);
P3 p3a2 = new P3(p2a.m_vertex.m_right,barrier, mesh);
P3 p3b1 = new P3(p2b.m_vertex.m_left,barrier, mesh);
P3 p3b2 = new P3(p2b.m_vertex.m_right,barrier, mesh);
p3a1.start();
p3a2.start();
p3b1.start();
p3b2.start();
barrier.await();

//Test the tree
TestTree tester = new TestTree();
boolean result = tester.test(S,0,4);
if(result==true)System.out.println("OK");
} catch (InterruptedException | BrokenBarrierException e) {
e.printStackTrace();
}
}
}
```

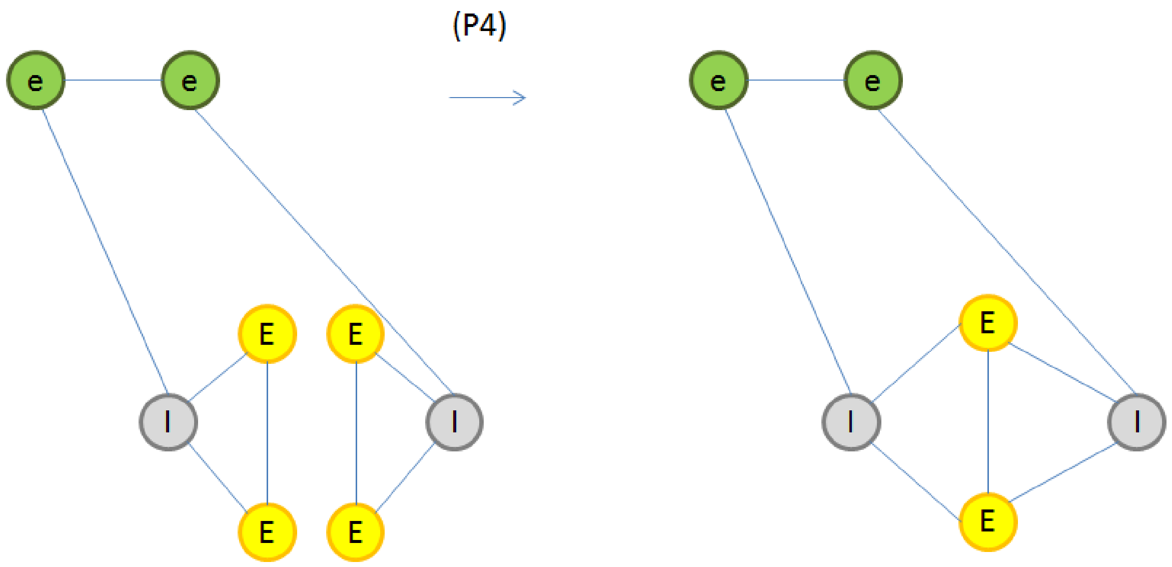
Please use similar mechanism (e.g. extend the provided JAVA code) to generate concurrently the recursive CP-graph.

Please use the following productions:

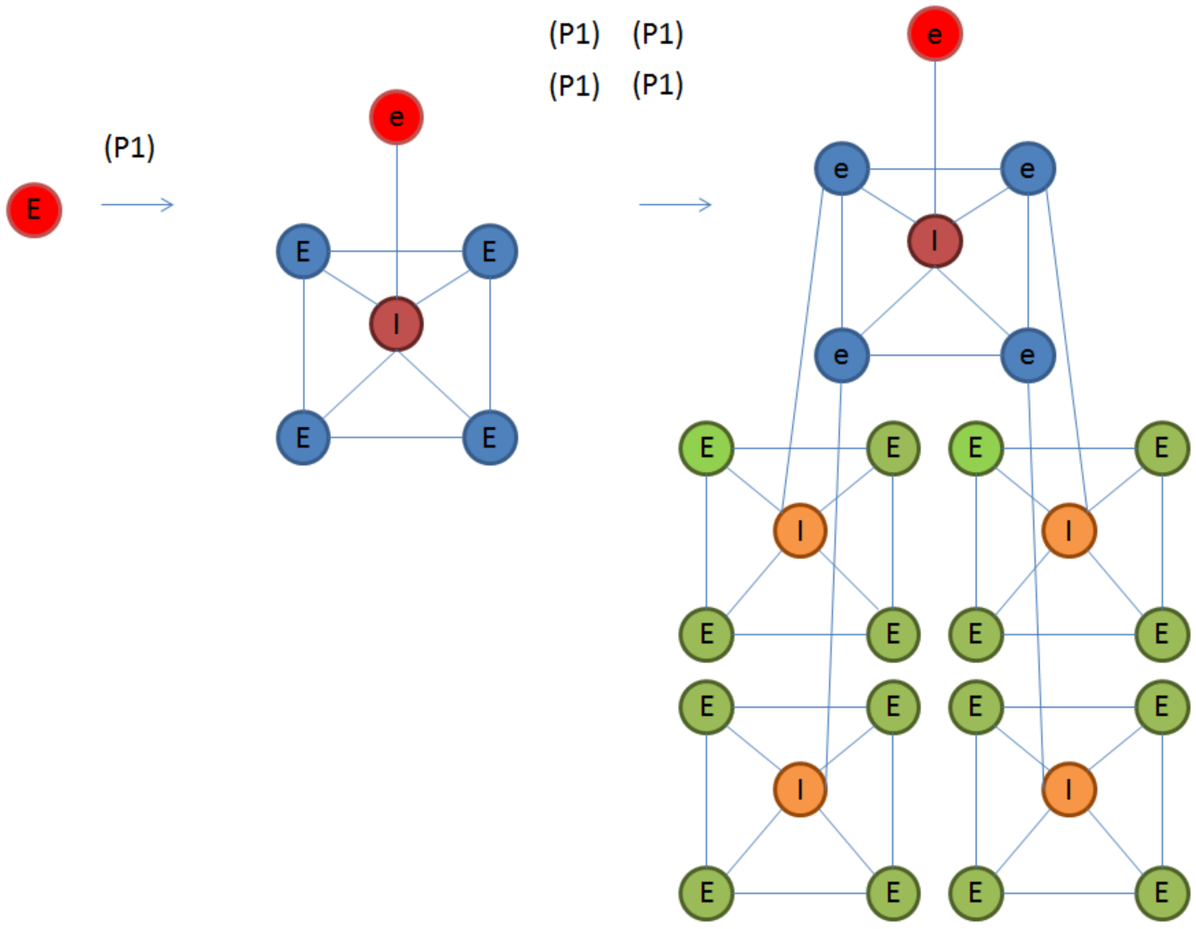



e
e
e
e
(P2)
e
e
e
e
I
I
E
E
E
E
I
I
I
I
E
I
I


(P3)
e
e
I
e
e
I
E
E
I
I
E
E
I
I
E

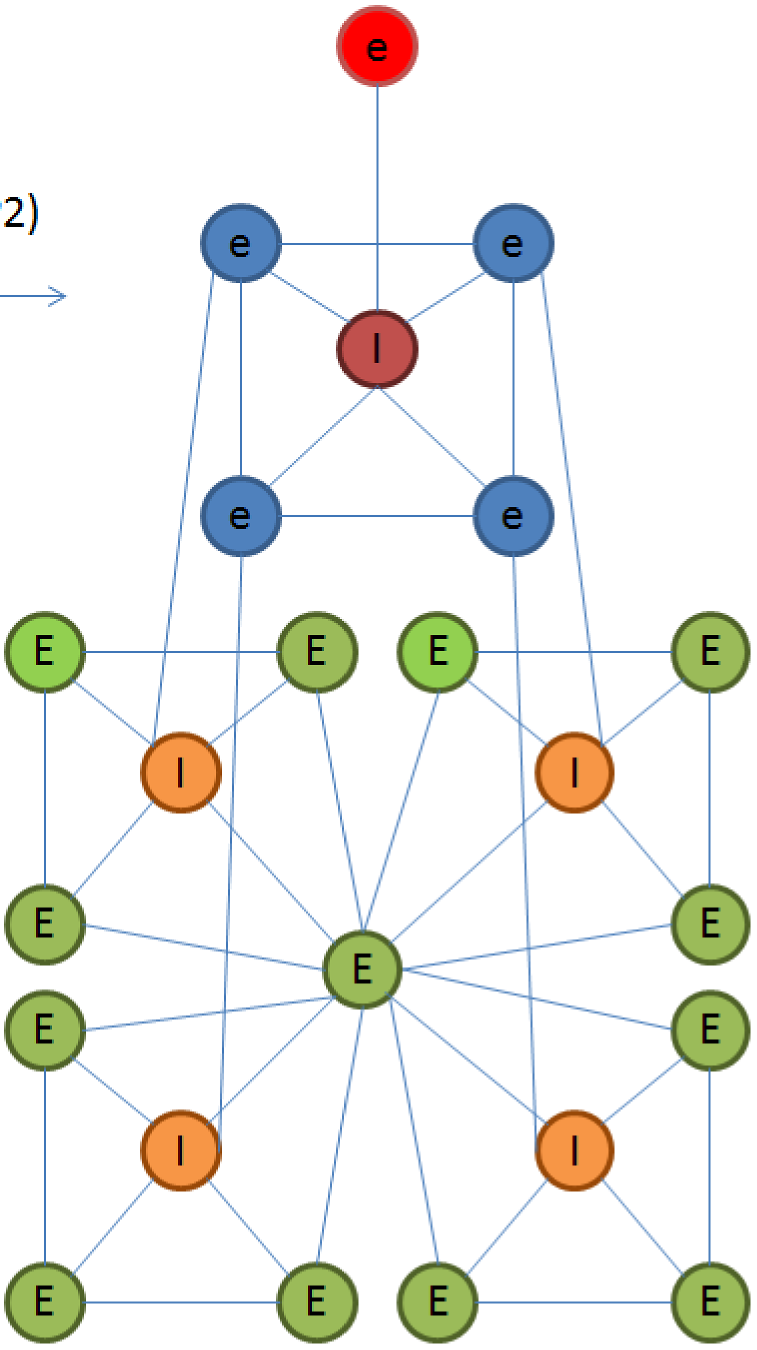


Please test it on the following derivation using maximum possible concurrency:



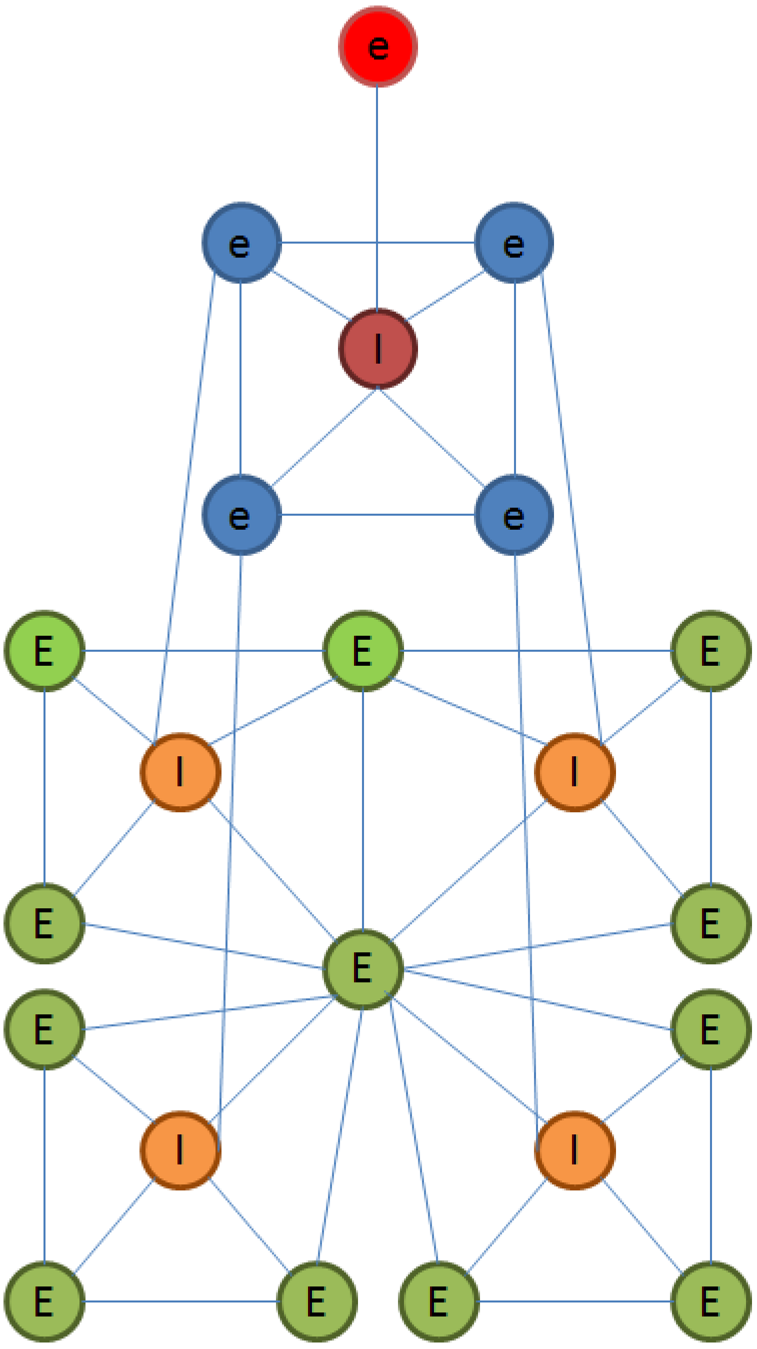
(P2)

$\longrightarrow$

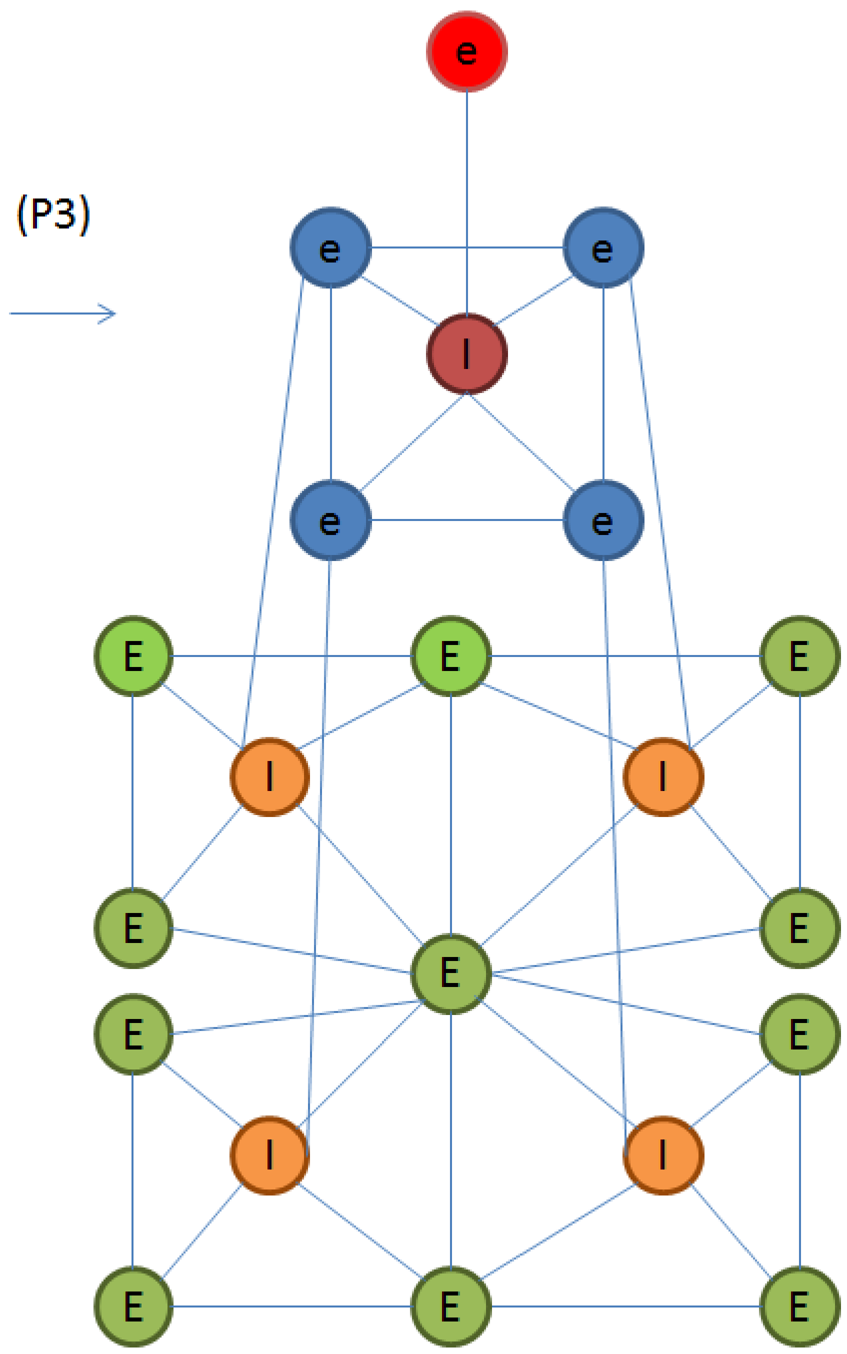


(P3)

$\longrightarrow$

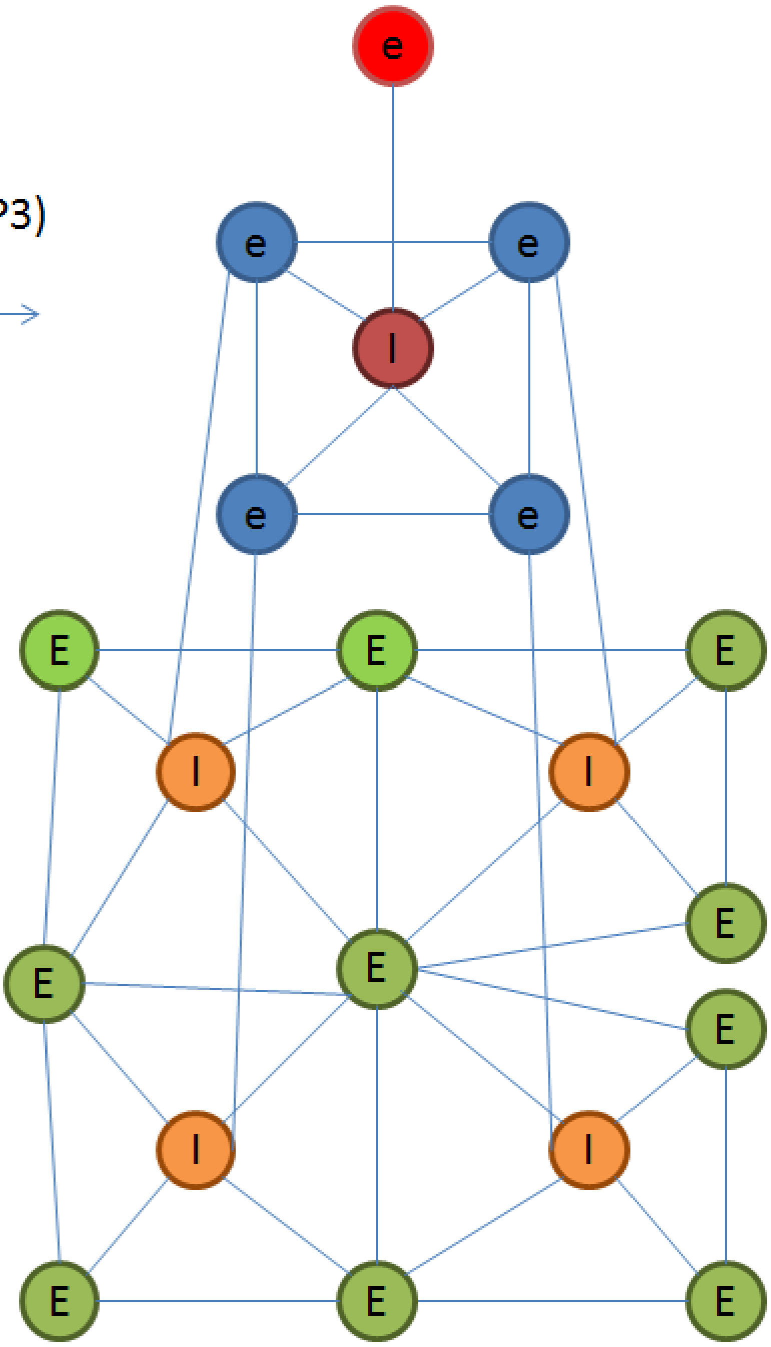


(P3)



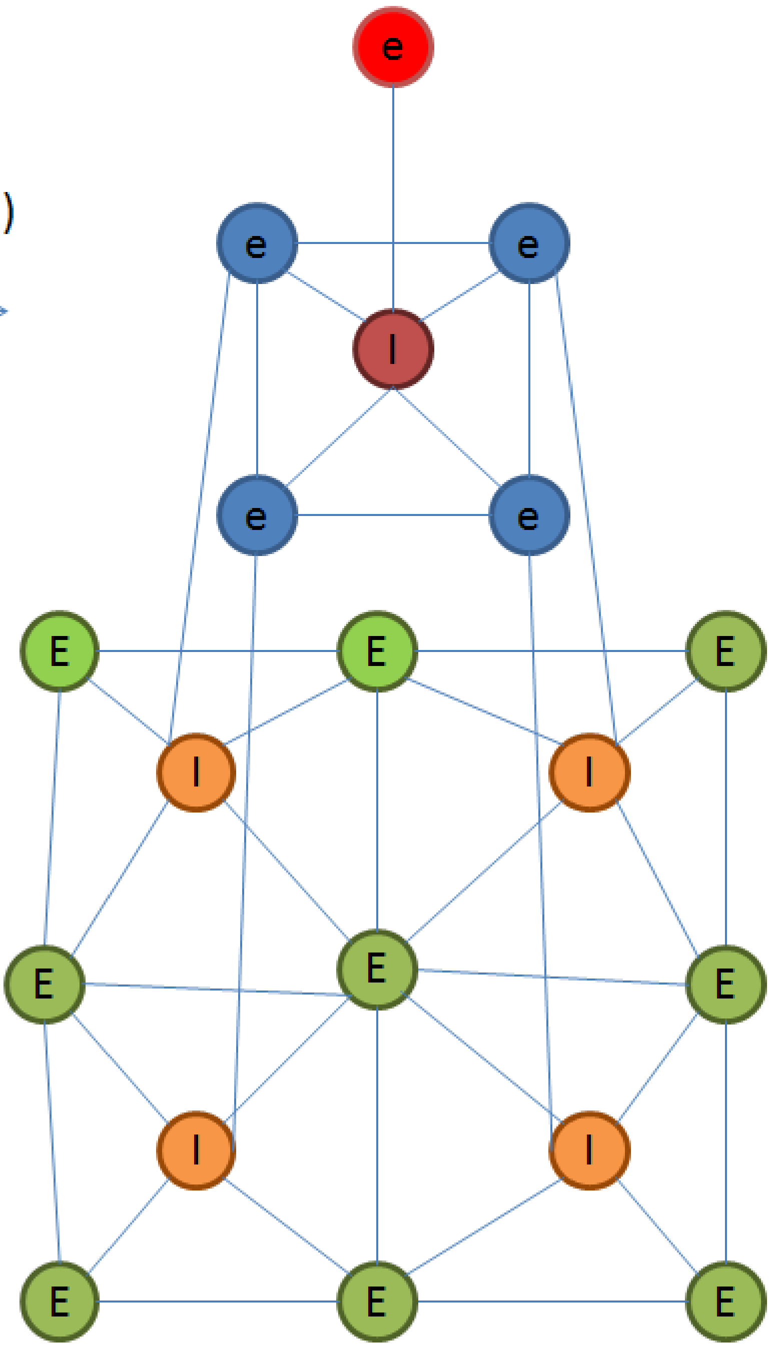
(P3)

$\longrightarrow$



(P3)

$\longrightarrow$

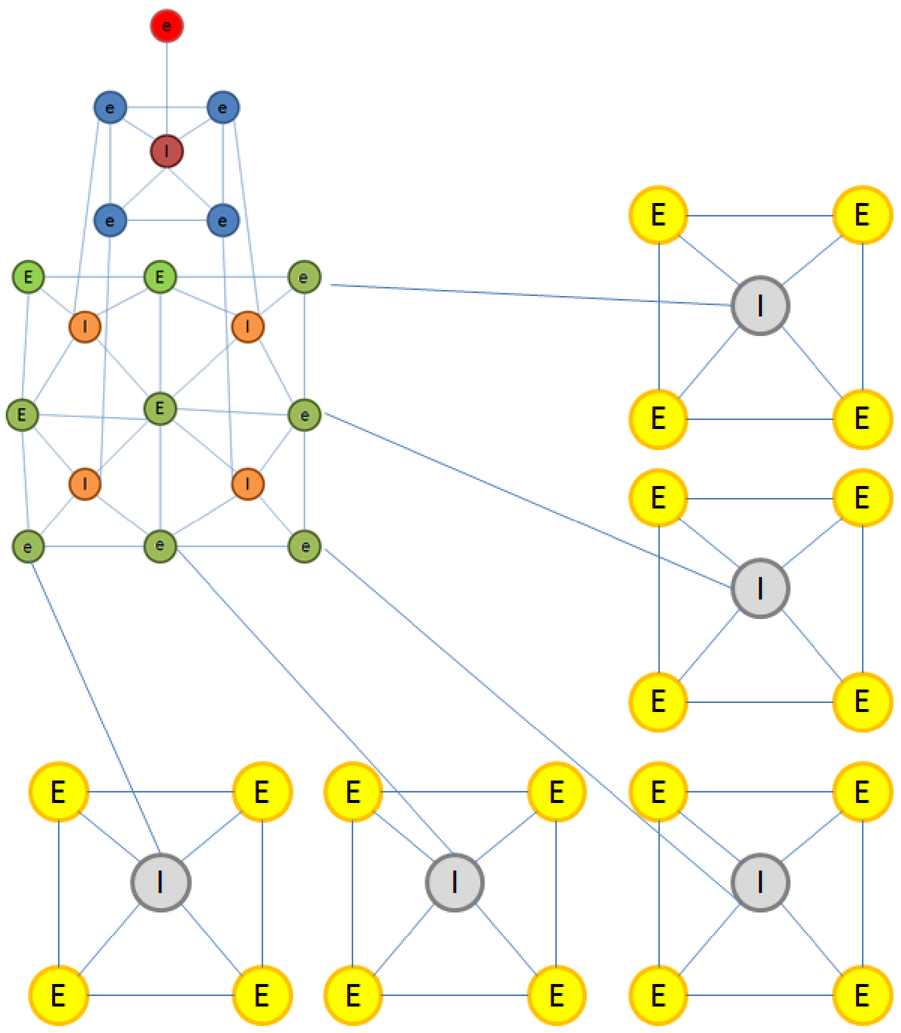


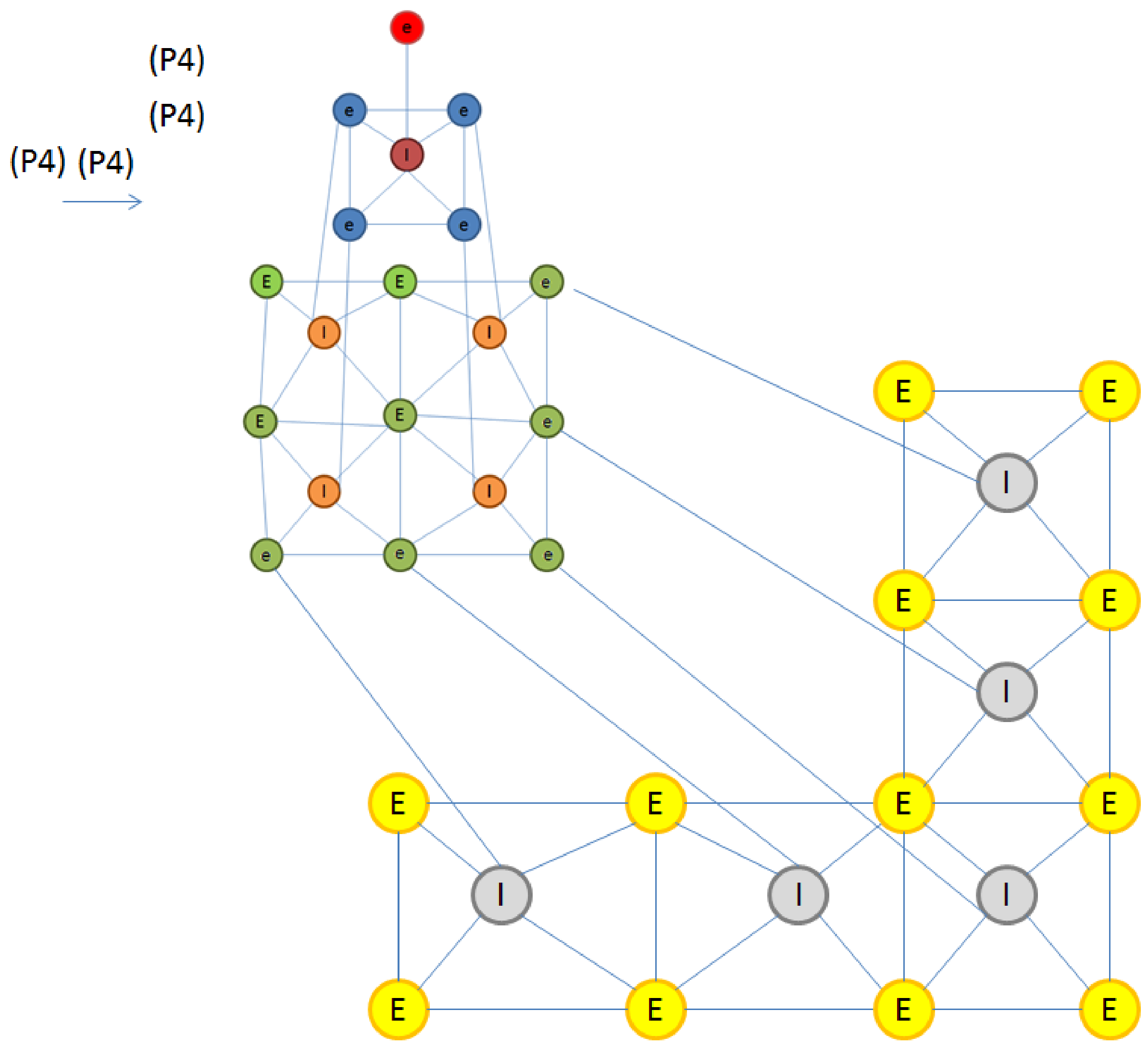
(P1)

(P1)

(P1) (P1) (P1)

$\longrightarrow$





Roles

Student 1. Implementation of the graph grammar production (P1)

Student 2. Design and implementation of the unit test for graph grammar production (P1)

Student 3. Implementation of the graph grammar production (P2)

Student 4. Design and implementation of the unit test for graph grammar production (P2)

Student 5. Implementation of the graph grammar production (P3)

Student 6. Design and implementation of the unit test for graph grammar production (P3)

Student 7. Implementation of the graph grammar production (P4)

Student 8. Design and implementation of the unit test for graph grammar production (P4)

Student 9. Implementation of the visualization tool for graph

Student 10. Design and implementation of integration tests for the visualization tool

Student 11. Storing graphs and derivations to files

Student 12. Design and implementation of integration tests for input/output

Deadlines:

Brain Storm March 8

Student 1 Student 2 Student 3 Student 4 March 15

Student 5 Student 6 Studnet 7 Student 8 March 30

Student 9 Student 10 Student 11 Student 12 May 5

All students - presentation of the derivation presented in this document May 17

All students - presentation of the corrections May 24