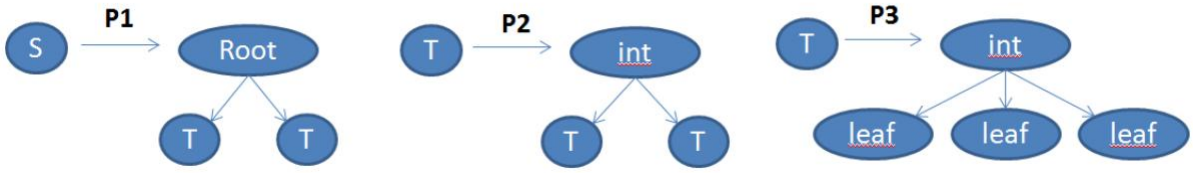
In the exemplary JAVA code

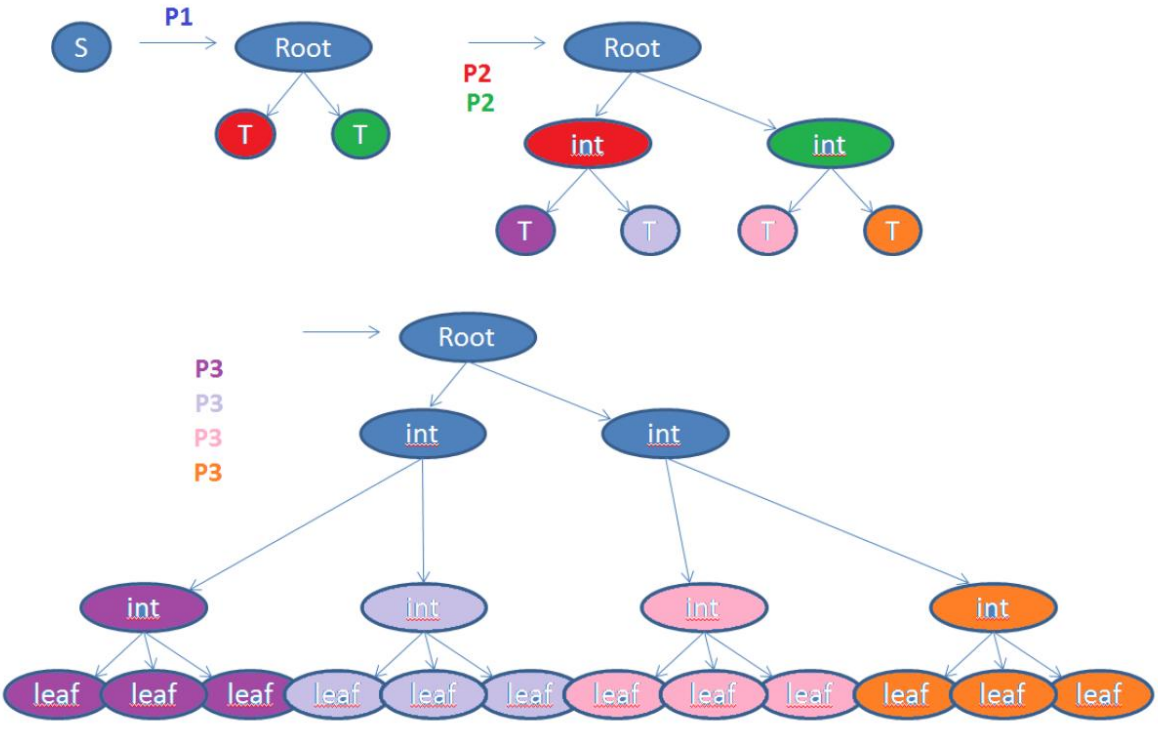
home.agh.edu.pl/paszynsk/GG

AdaptiveMeshGenerator_Source.zip

we have an Executor class where the tree grammar is scheduled to construct binary tree using the following productions



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



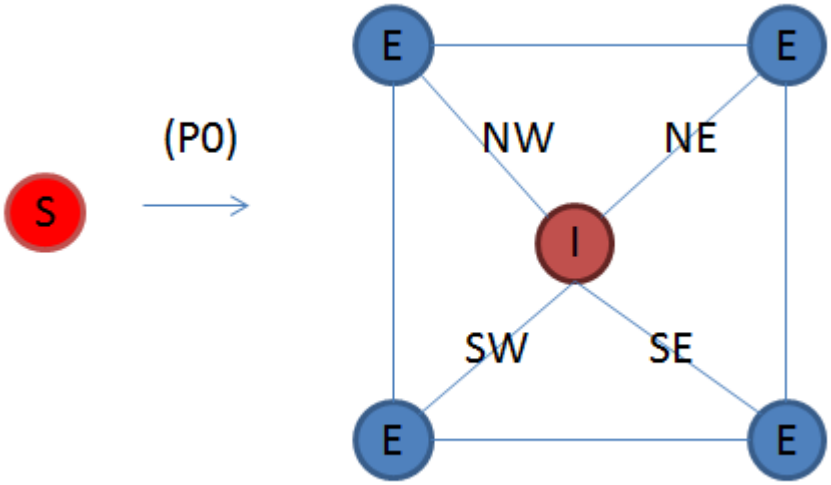
The concurrent scheduling of the productions is managed in the Executor class

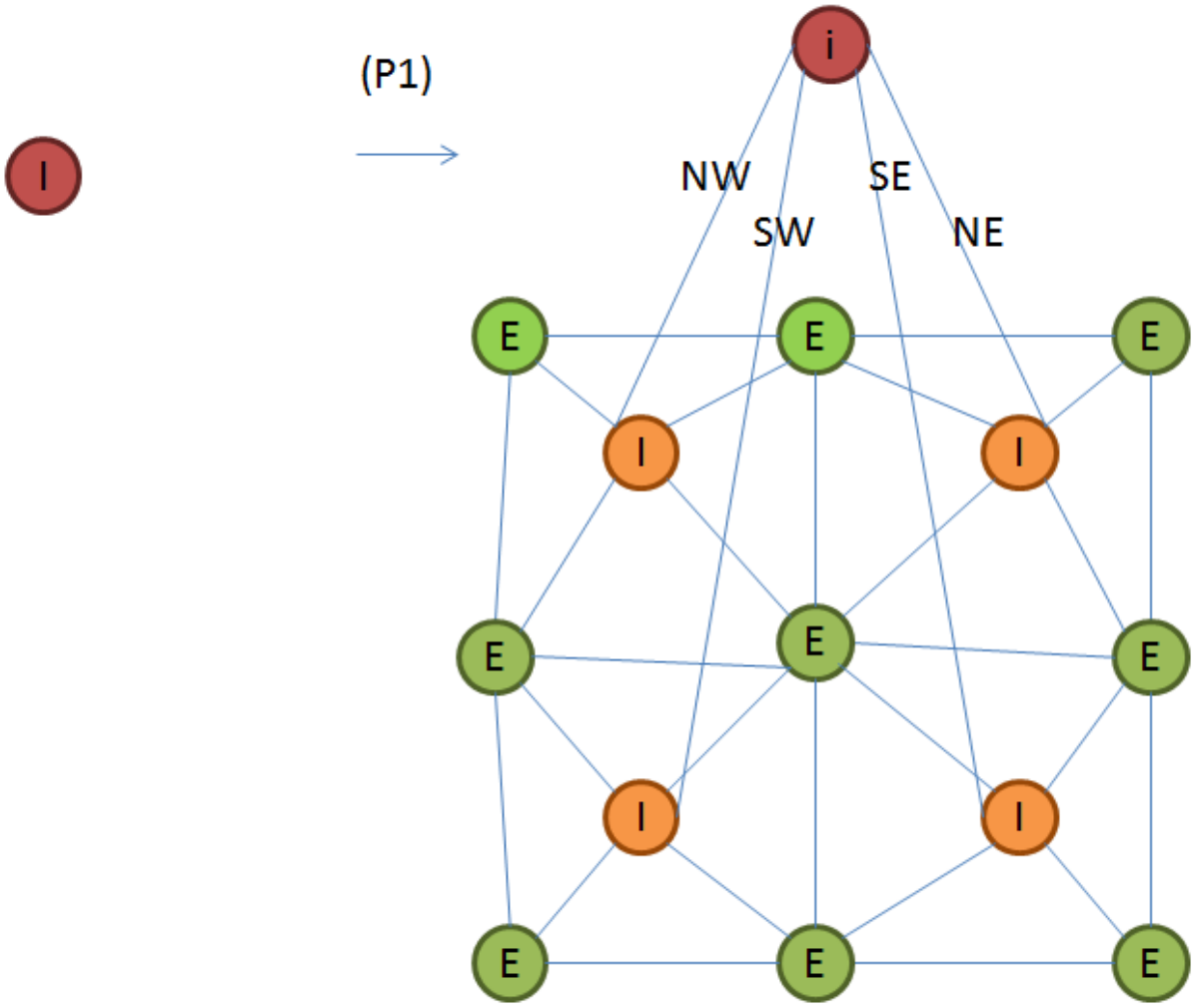
```
class Executor extends Thread {
public synchronized void run() {

// BUILDING TREE

try {
//...

//[(P1)]
CyclicBarrier barrier = new CyclicBarrier(1+1);
```

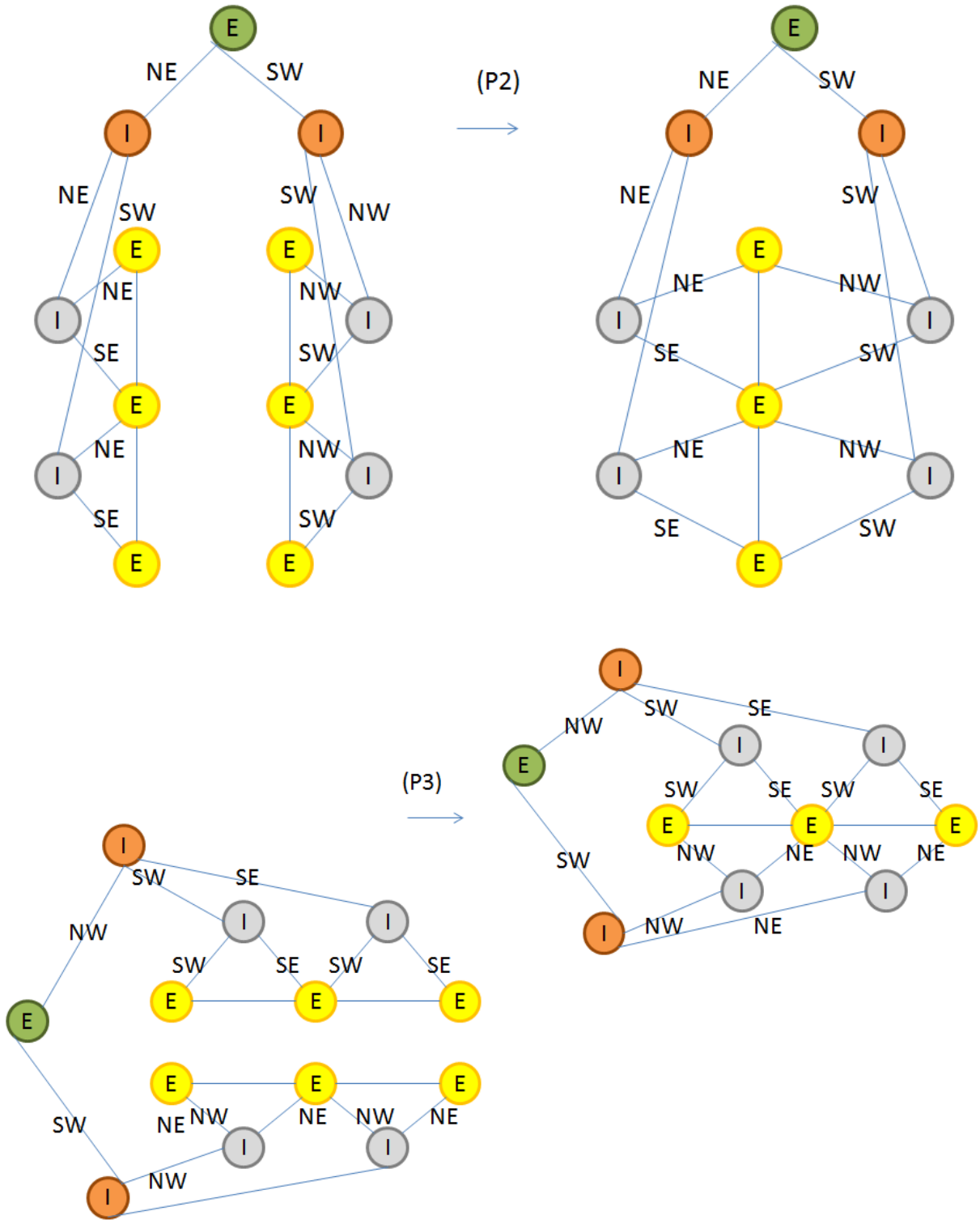
```
P1 p1 = new P1(S,barrier, mesh);
p1.start();
barrier.await();

//[(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.

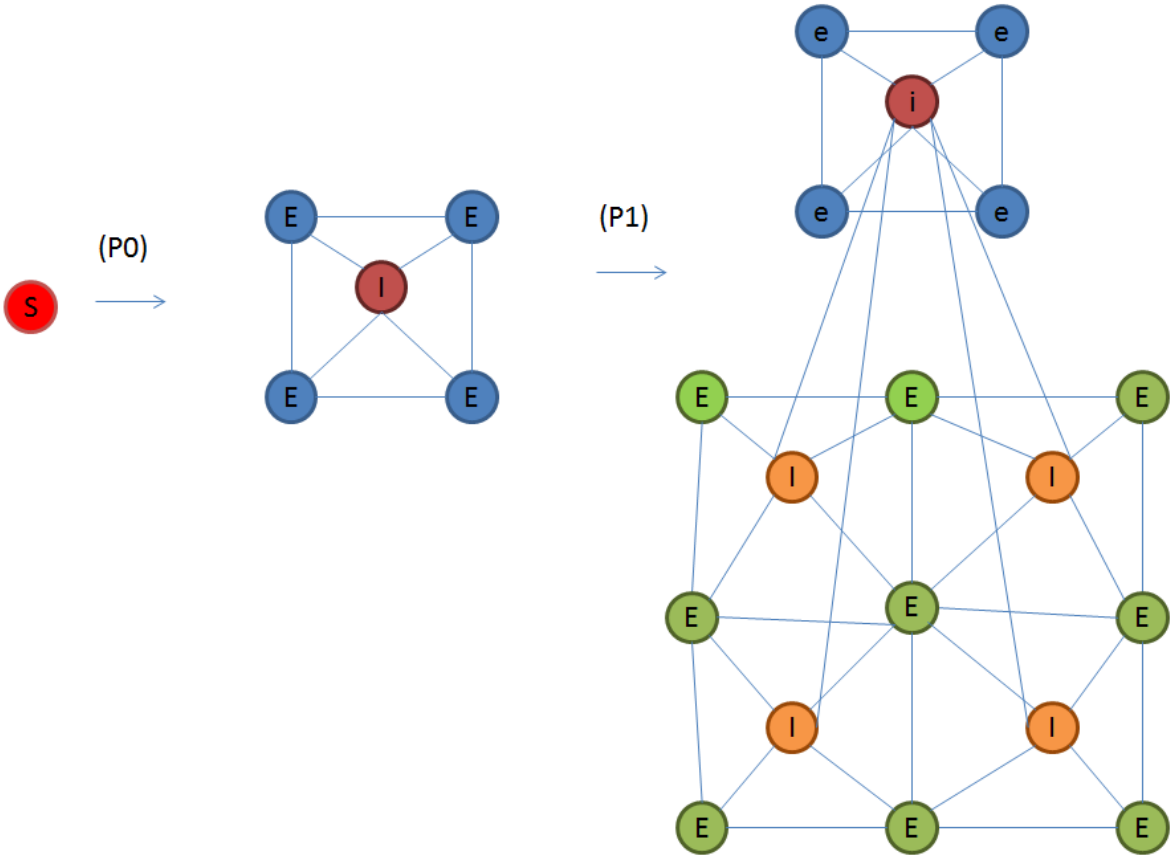
Please use the following productions:

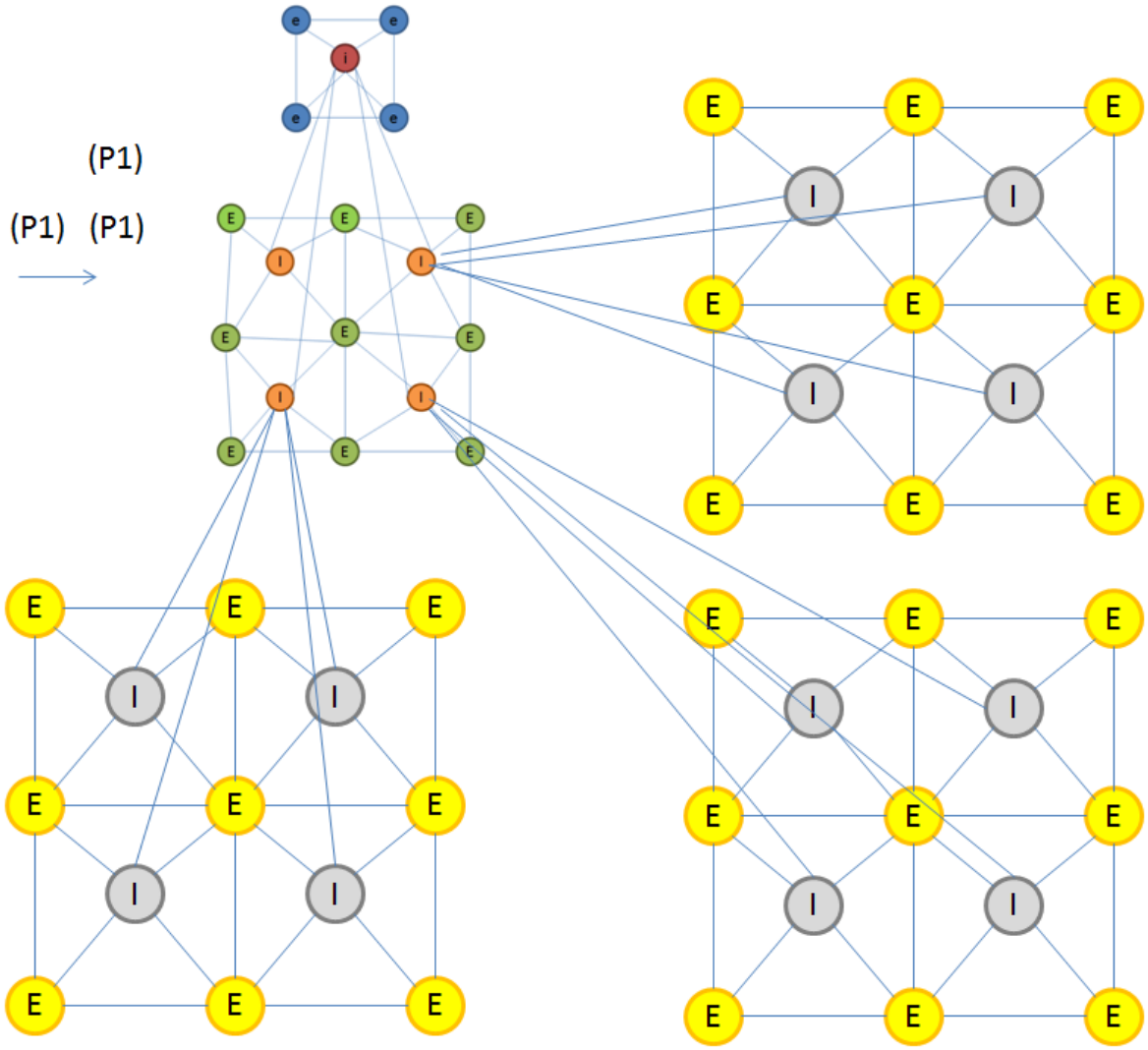



I
(P1)
i
NW
SW
SE
NE
E
E
E
I
I
E
E
E
I
I
E
E
E

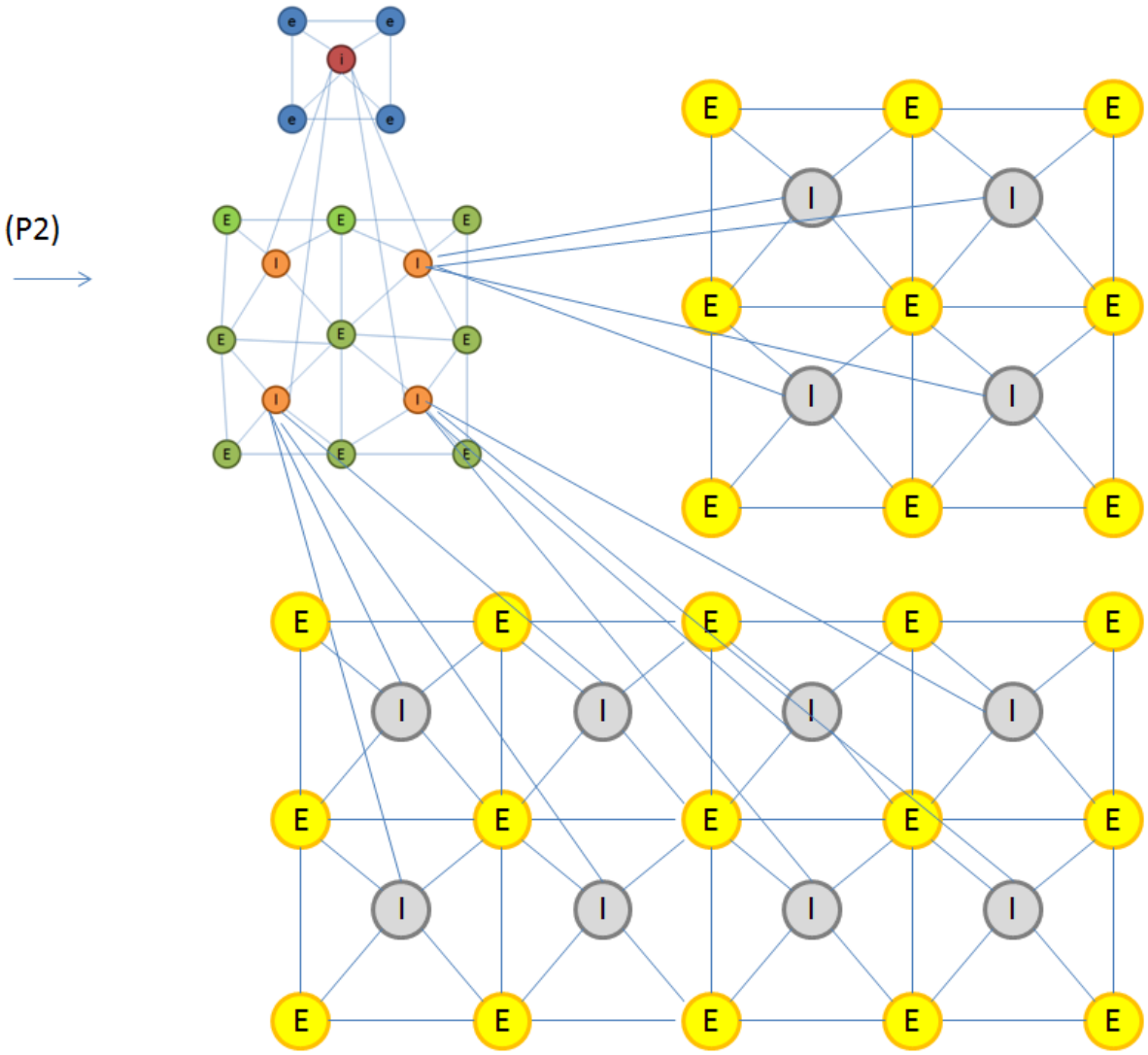


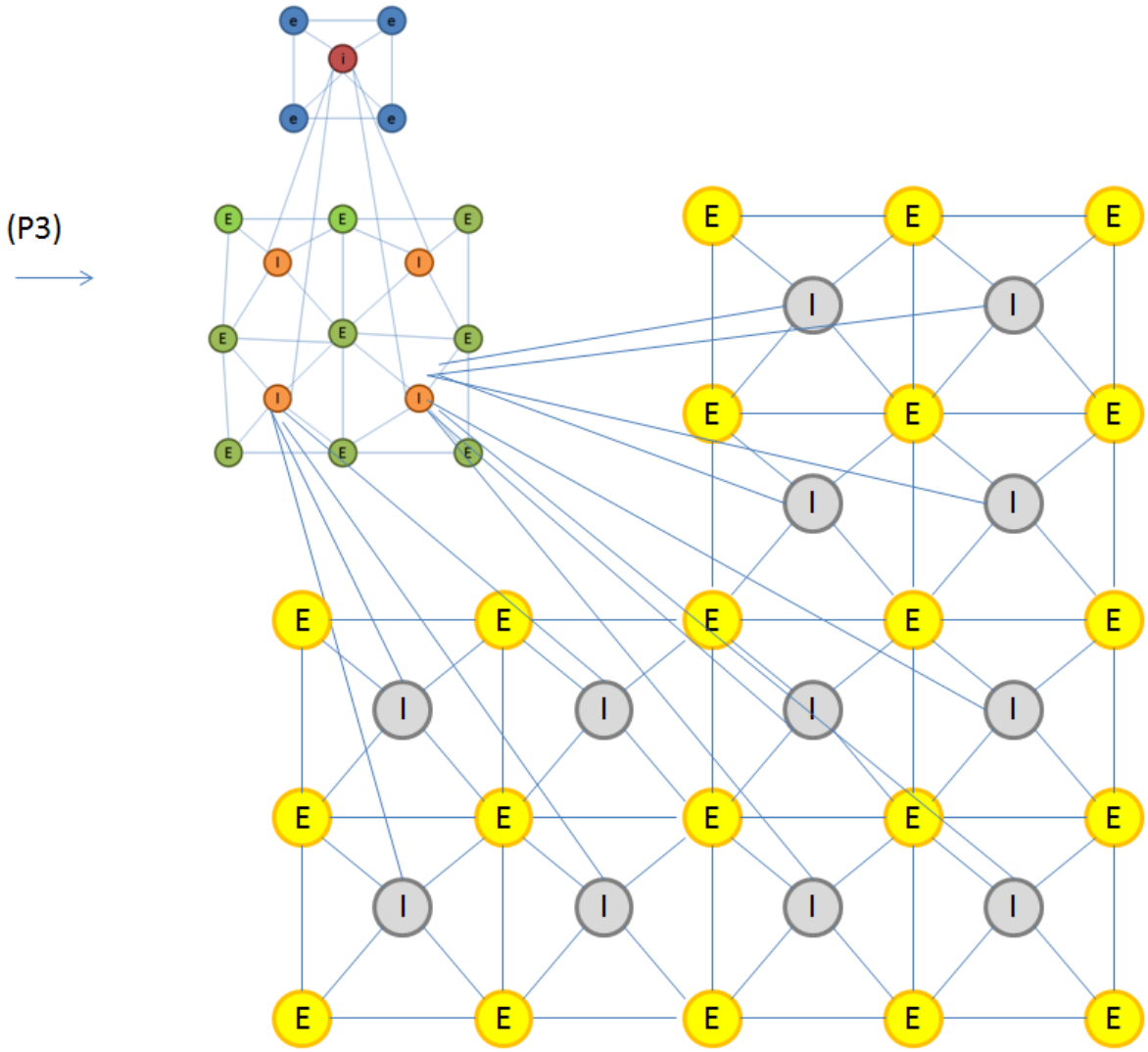
Please test it on the following derivation using maximum possible concurrency:


S
(P0)
E
E
I
E
E
(P1)
e
e
i
e
e
E
E
E
I
I
E
E
E
I
I
E
E
E


(P1)
(P1) (P1)
e
i
E
I

(P2)


e
e
i
e
e
E
E
E
I
I
E
E
E
I
I
E
E
E
E
E
E
E
E
E
I
I
E
E
E
I
I
E
E
E
E
E
E
E
E
E
I
I
E
E
E
I
I
E
E
E
E
E



Roles

Student 1. Implementation of the serialization of the data structure allowing for implementation of graph grammar productions and derivation process

Student 2. Design and implementation of tests of data structure and serialization

Student 3. Implementation of the graph grammar production (P0)

Student 4. Design and implementation of the unit test for graph grammar production (P0)

Student 5. Implementation of the graph grammar production (P1)

Student 6. Design and implementation of the unit test for graph grammar production (P1)

Student 7. Implementation of the graph grammar production (P2)

Student 8. Design and implementation of the unit test for graph grammar production (P2)

Student 9. Implementation of the graph grammar production (P3)

Student 10. Design and implementation of the unit test for graph grammar production (P)

Student 11. Implementation of the visualization tool for graph

Student 12. Design and implementation of integration tests for the visualization tool

Deadlines:

And all the students agrees on data structure -- April 1 (MP)

Student 3 Student 4 Student 5 Student 6 -- May 12 (MP)

Student 7 Student 8 Student 9 Student 10 -- May 19 (MP)

Student 1 and Student 2 and Student 11 Student 12 -- May 26 (ML)

All students - presentation of the derivation presented in this document -- June 2 (ML)

All students - presentation of the corrections -- June 16 (MP)