Testing Model Tranformations

Bottom Up and Top Down

Amr Al Mallah

Testing Model Transformation

1. Model Transformations frameworks :
   1.1 MoTif in context of Traffic 2 Petri Net.

2. Testing this Model transformation :
   2.1. T-Unit approach (half modular) :
      2.1.1 : Matching problem (in general and in Atom3 models).
      2.1.2 : Criteria matching (building + compiling + using T-Unit)

3. Modeling the model transformation framework:
   3.1 Overall testing blocks.
   3.2 Input model generator block.
   3.3 Result acceptor block.
   3.4 The Analyzer/Invoker block

Testing Model Transformation

Issues/ and comments:

1. Unit Testing approach i.e test cases are already specified.
2. Biggest issue is model comparison.
3. Integration with Sagar's work .

Example : From Traffic to Petri Nets

Input:
Traffic System formalism:
- Generators
- Road Segments (Capacity)
- In/out Ports
- Collectors

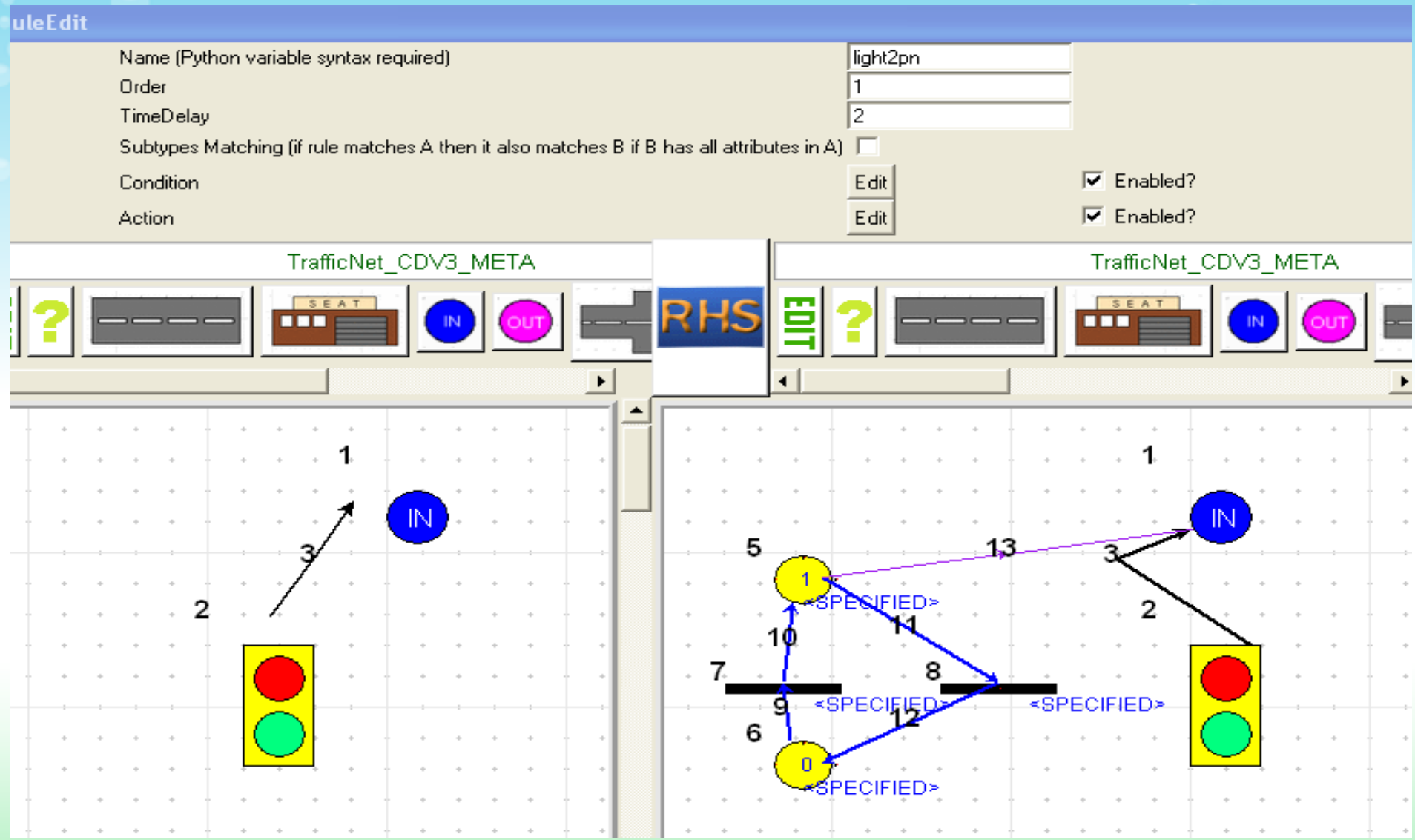
Output:
Petri net formalism:
- Places with tokens
- Transitions

Model to Model transformation between different Meta Models !

Traffic 2 PN in GG rules

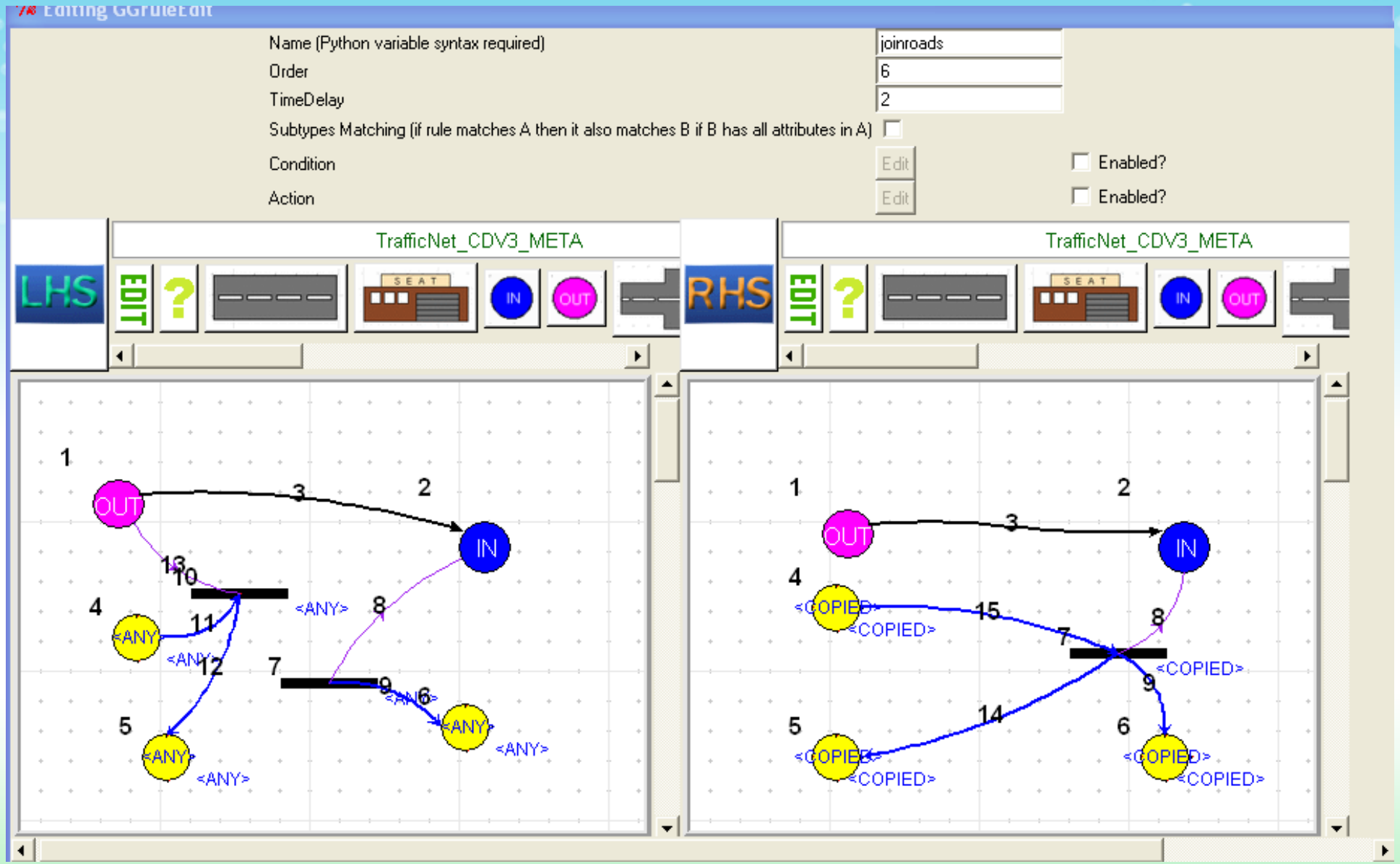
1. Defined the transformation as a bunch of rules executed in a specific sequence.
2. These rules involve a lot of intermediary steps :
    1. Add a Petri net Light to each traffic light.
    2. Add a Petri net Generator to each traffic generator.
    3. ..
    4. ..
    5. connect Petri net Road segments
    6. ..
    7. ..
    8. remove traffic light .
    9. ..
    10. ..

Traffic 2 PN in GG rules



Add a PN light to each traffic light

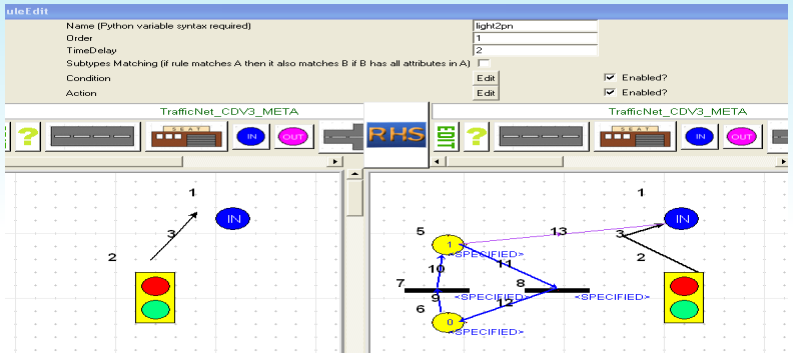
Traffic 2 PN in GG rules



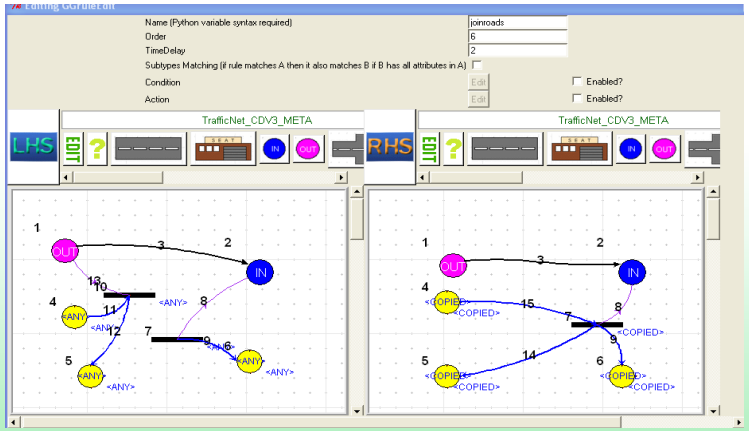
Join PN road segments

Traffic 2 PN in MoTif

1. Rule Compiler :



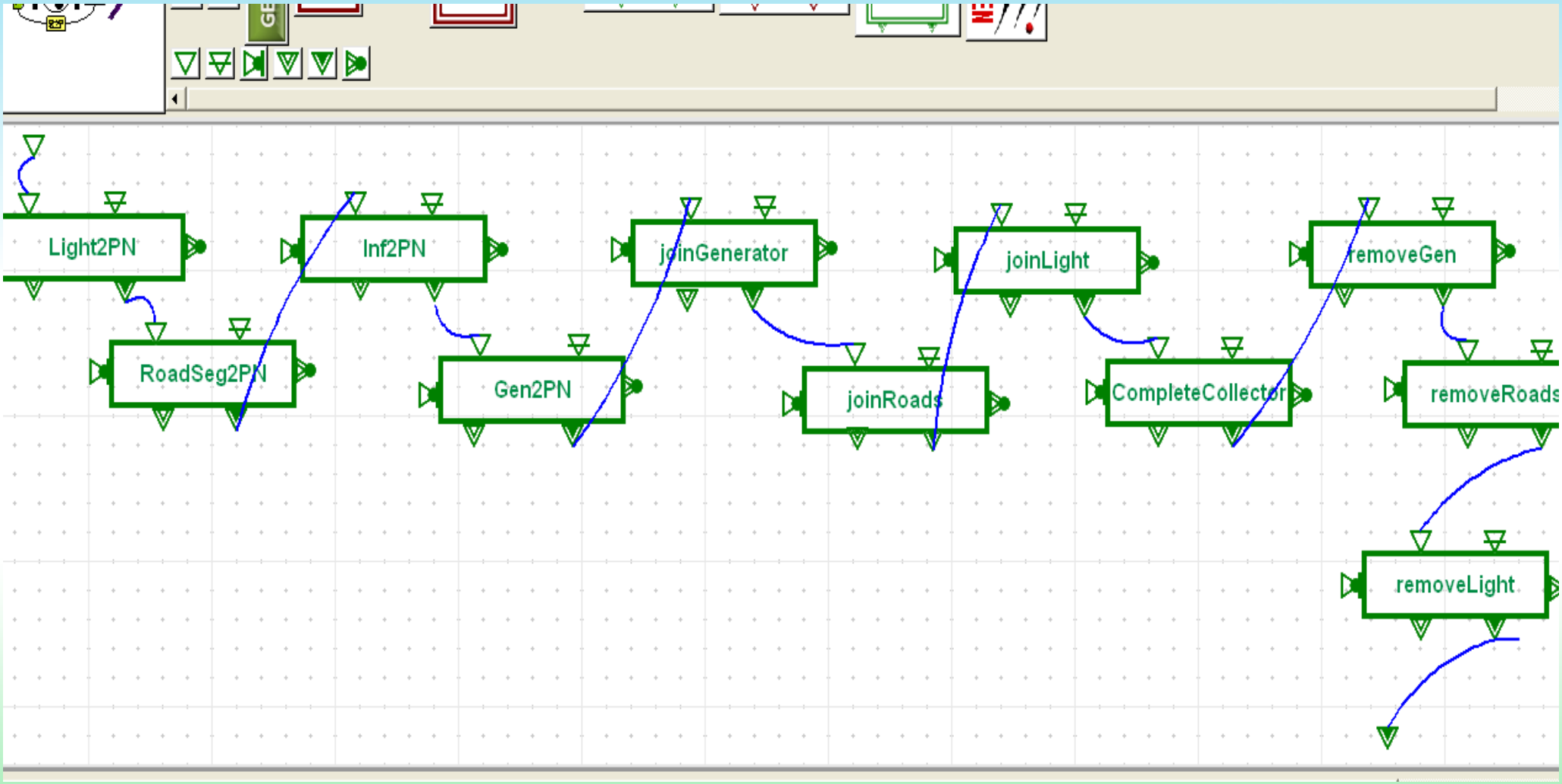
Rule 1



Rule 2

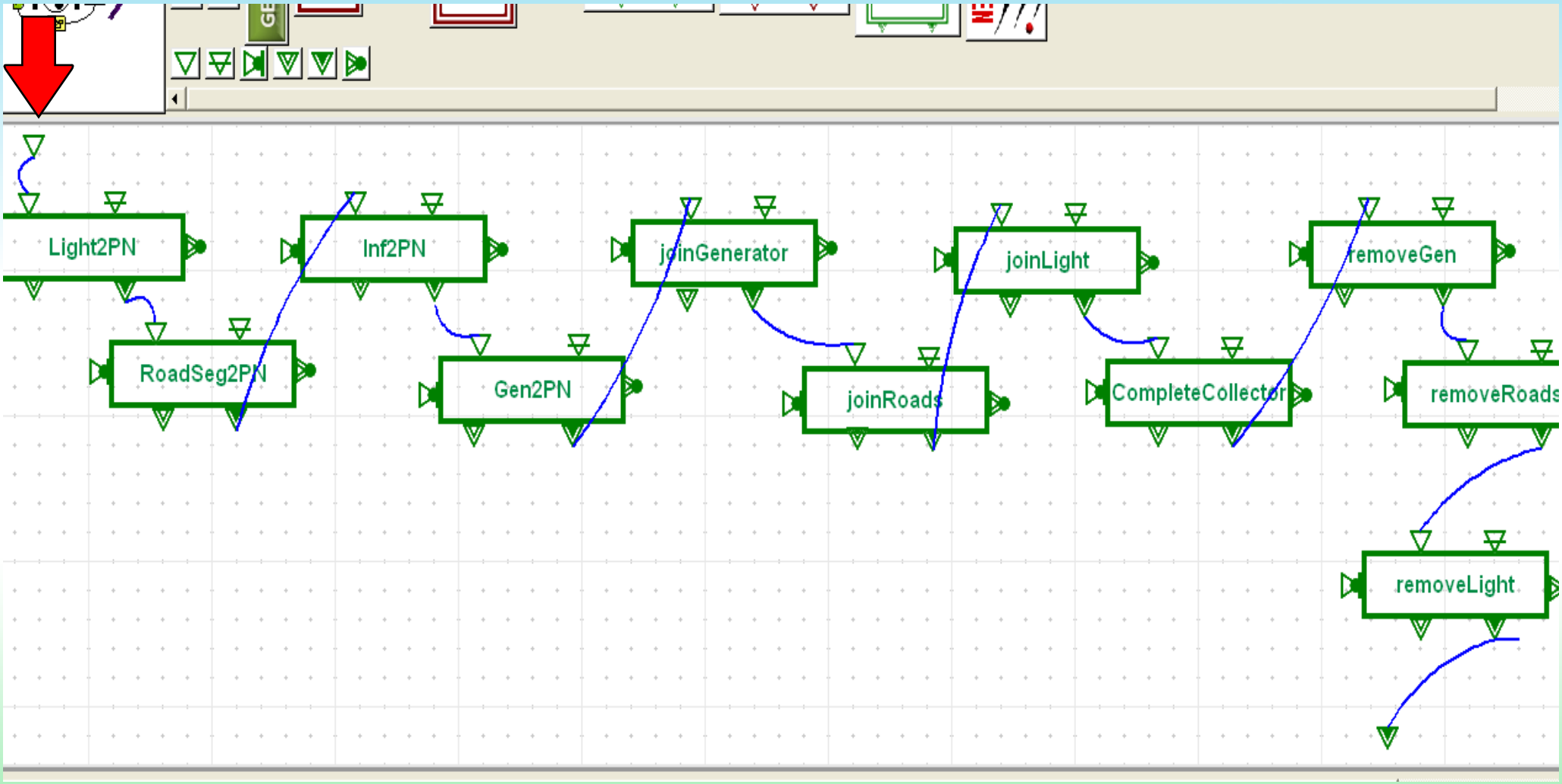
Traffic 2 PN in MoTif

2. Traffic 2 PN MoTif Model :



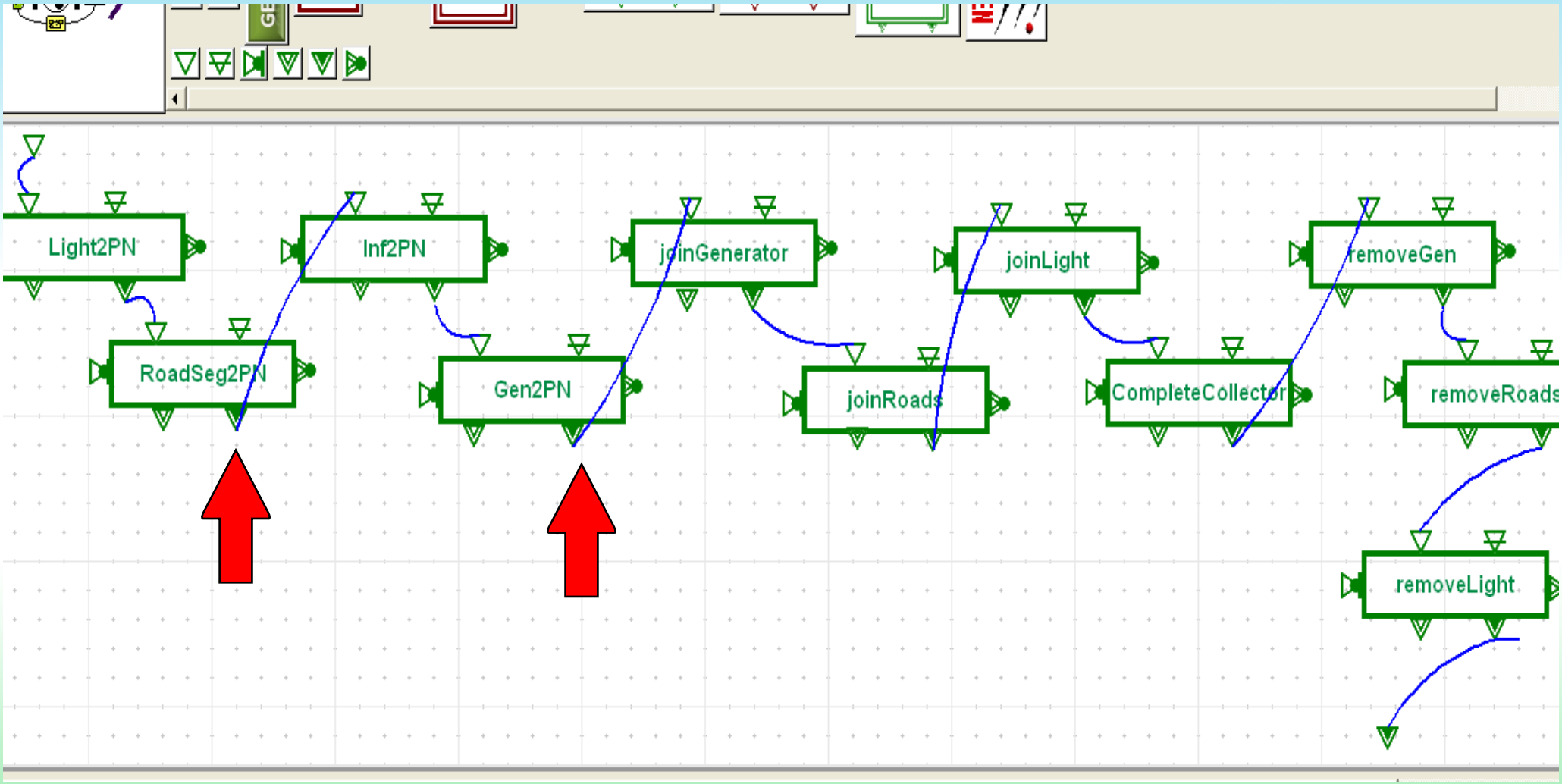
Traffic 2 PN in MoTif

2. Traffic 2 PN MoTif Model :



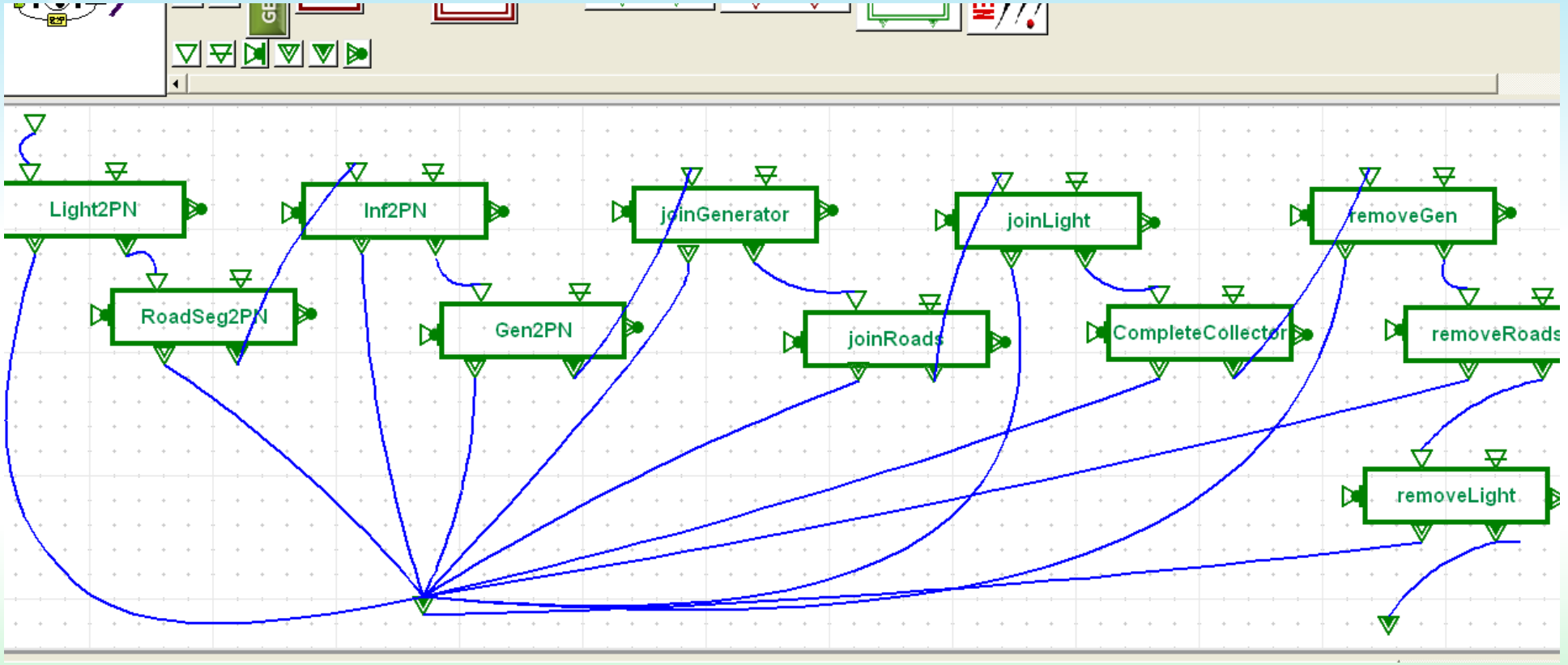
Traffic 2 PN in MoTif

2. Traffic 2 PN MoTif Model :



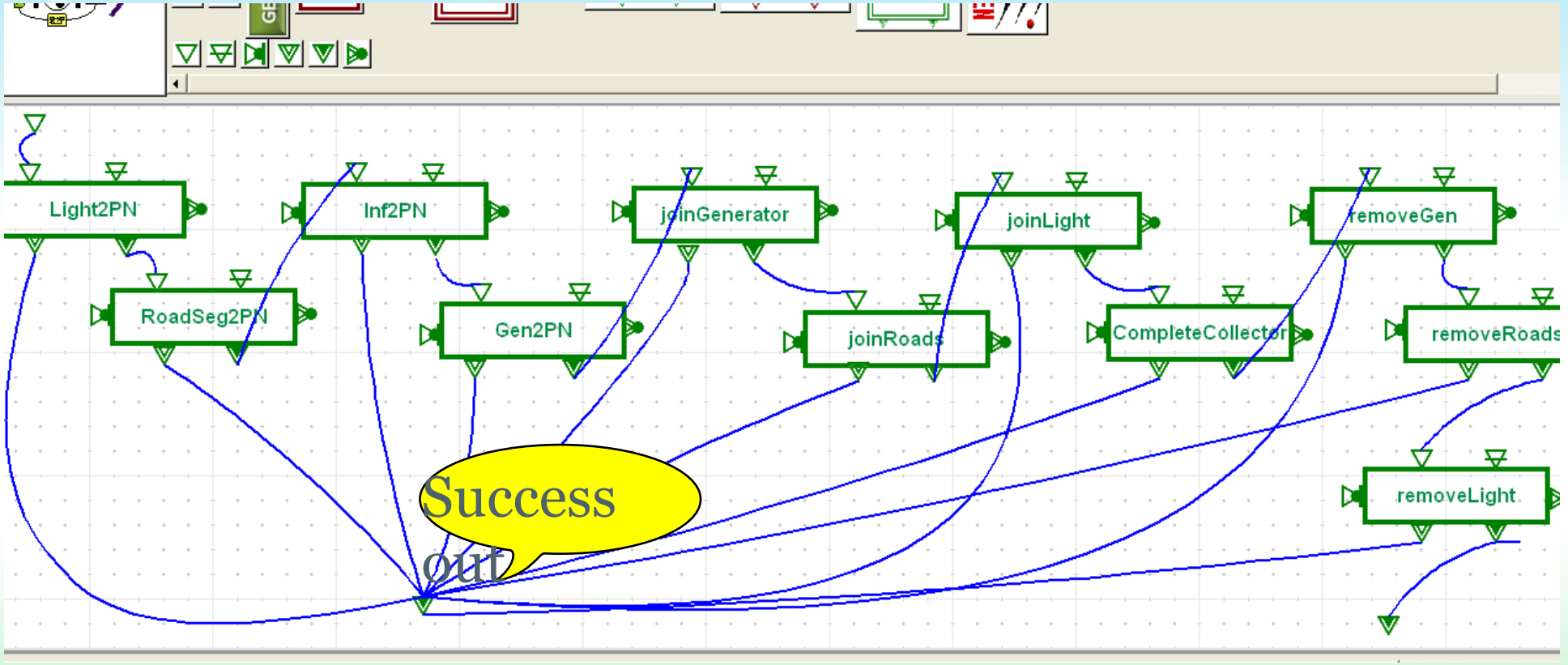
Traffic 2 PN in MoTif

2. Traffic 2 PN MoTif Model :



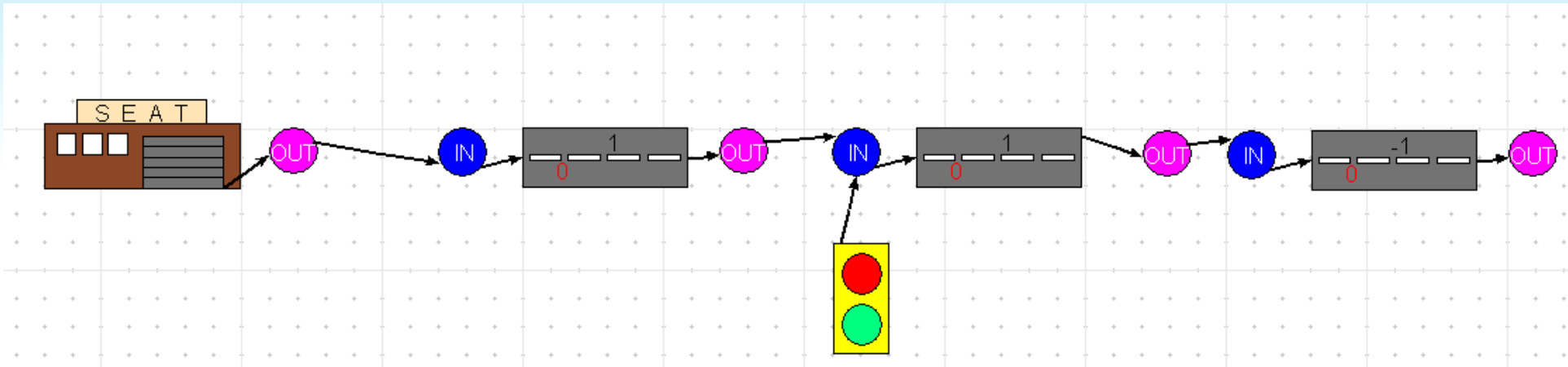
Traffic 2 PN in MoTif

2. Traffic 2 PN MoTif Model :



Traffic 2 PN in MoTif

3. Compile an example input model and link to the generated MoTif environment .



Traffic 2 PN in MoTif

4. Execute the transformation

Problems :

1. NAC or visit nodes only once
2. Different Meta models

```
---------- Transforming graph ----------
Light2pn
<ATOM3.Graph.Graph instance at 0x01858DF0>
Roadsegment2pn
<ATOM3.Graph.Graph instance at 0x01907C60>
Gen2pn
<ATOM3.Graph.Graph instance at 0x01939418>
Joingenerator
<ATOM3.Graph.Graph instance at 0x0192D990>
Joingenerator
<ATOM3.Graph.Graph instance at 0x019C11E8>
Removegen
<ATOM3.Graph.Graph instance at 0x01853558>
Removeroads
<ATOM3.Graph.Graph instance at 0x01ACE0D0>
Roadsegment2pn
<ATOM3.Graph.Graph instance at 0x019175F8>
Joingenerator
<ATOM3.Graph.Graph instance at 0x01B54BC0>
Removeroads
<ATOM3.Graph.Graph instance at 0x018589E0>
Roadsegment2pn
<ATOM3.Graph.Graph instance at 0x01B3E120>
Completecollector
<ATOM3.Graph.Graph instance at 0x01B486C0>
Removeroads
<ATOM3.Graph.Graph instance at 0x01866580>
Removelight
<ATOM3.Graph.Graph instance at 0x019A8EE0>
>>>
```

Bottom up testing framework : TUnit

Main Items Needed for it to be usable:

1. Describing the input, output model pairs
2. Execute the transformation and collect result
3. Compare Result with expected output model.
4. Collect meaningful results and a produce a report.

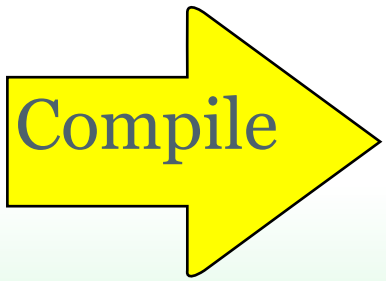
Bottom up testing framework : TUnit

Main Items Needed for it to be usable:

1. Describing the input, output model pairs
2. Execute the transformation and collect result
3. Compare Result with expected output model.
4. Collect meaningful results and a produce a report.

Model Input in Atom3 / Other

Model Output in Atom3 / Other



Input 1

Output 1

Bottom up testing framework : TUnit

Main Items Needed for it to be usable:

1. Describing the input, output model pairs
2. Execute the transformation and collect result
3. Compare Result with expected output model.
4. Collect meaningful results and a produce a report.

Extended The User Block in MoTif to Remember the last tranformed graph it received

Bottom up testing framework : TUnit

Main Items Needed for it to be usable:

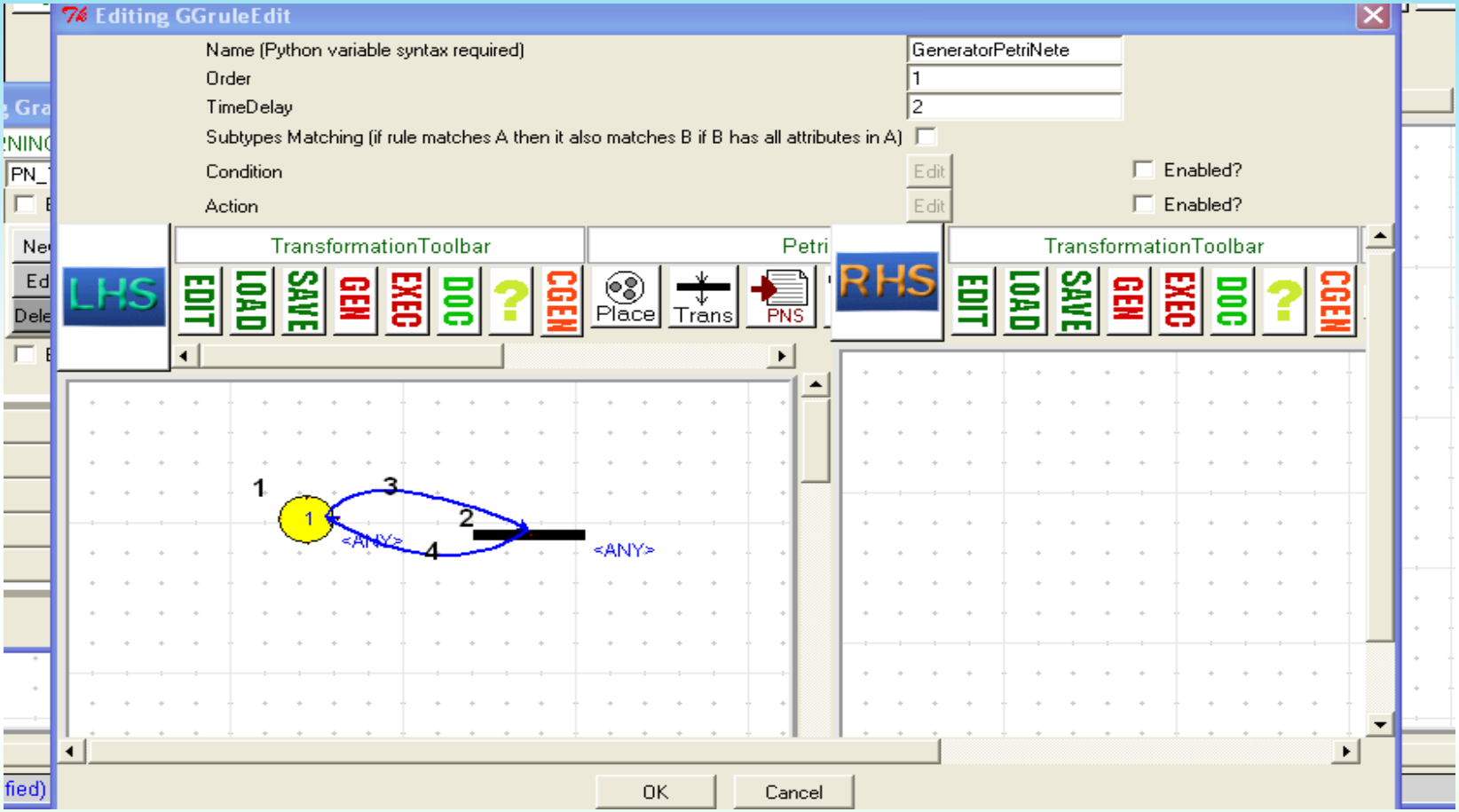
1. Describing the input, output model pairs
2. Execute the transformation and collect result
3. Compare Result with expected output model.
4. Collect meaningful results and a produce a report.

RoadSegment: name: tr_Foo	Equal ?	RoadSegment: name: tr_bar

Direct Matching is Dumb: Match Behavior
Not Semantics!! Yet !

Bottom up testing framework : TUnit

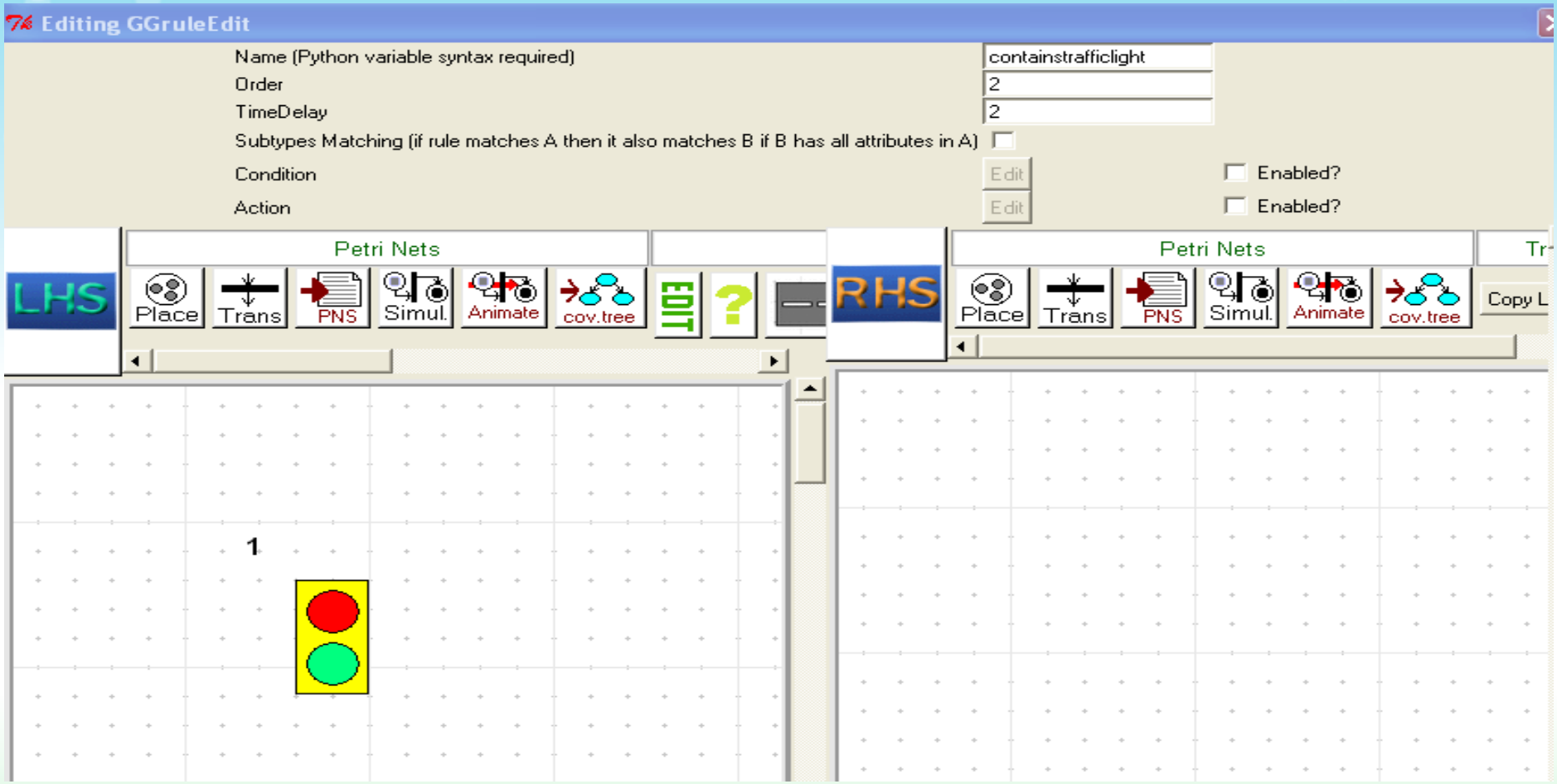
3. Compare Result with expected output model.



Criteria1 : Contains-PN-Generator

Bottom up testing framework : TUnit

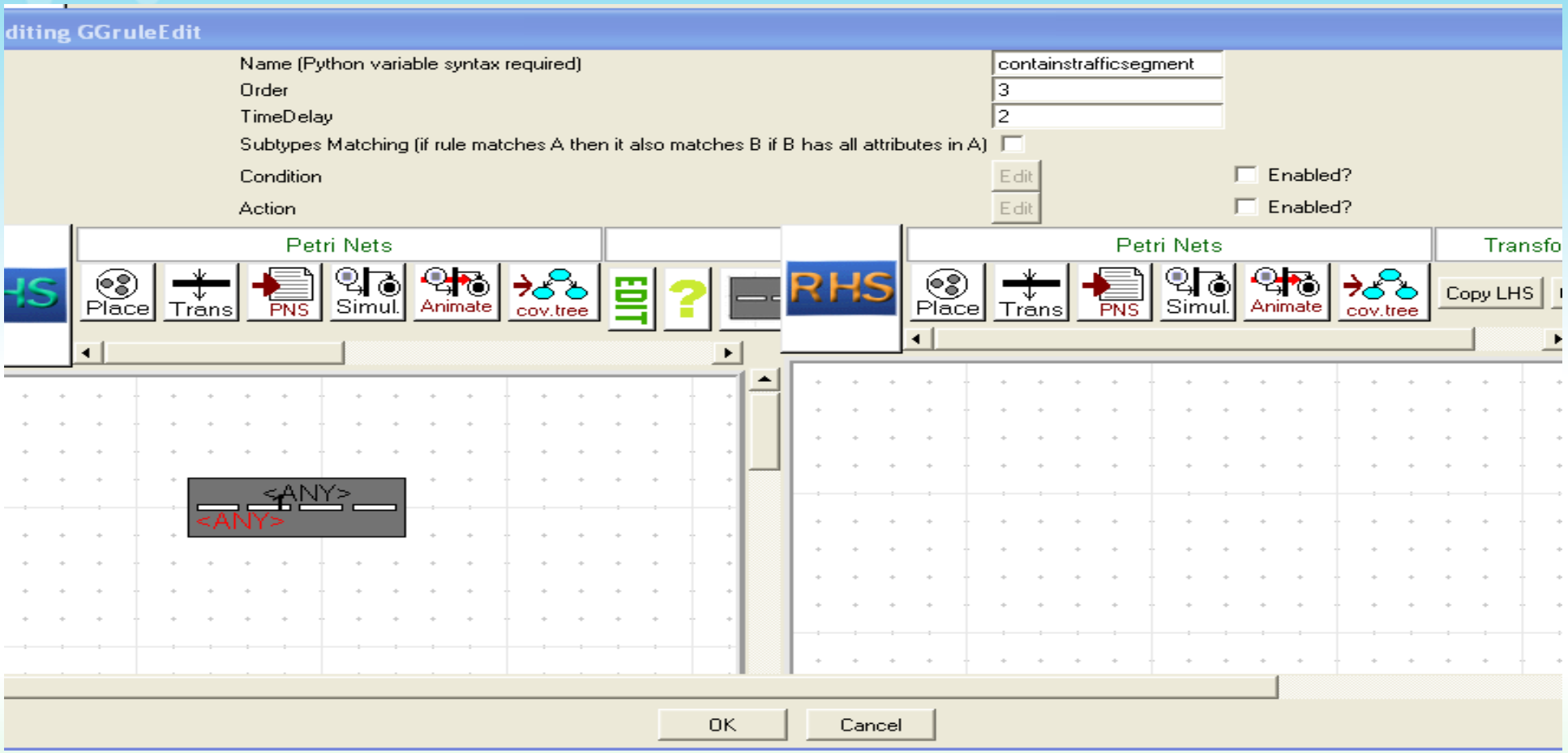
3. Compare Result with expected output model.



Criteria2 : Contains-Traffic-Light

Bottom up testing framework : TUnit

3. Compare Result with expected output model.



Criteria3 : Contains-Traffic-segment

Bottom up testing framework : TUnit

Main Items Needed for it to be usable:

1. Describing the input, output model pairs
2. Execute the transformation and collect result
3. Compare Result with expected output model.
4. Collect meaningful results and a produce a report.

RoadSegment:
name: tr_Foo

Has?

C -1

C -2

C -3

Direct Matching is Dumb: Match Behavior
Not Semantics!! Yet !

Bottom up testing framework : TUnit

Main Items Needed for it to be usable:

1. Describing the input, output model pairs
2. Execute the transformation and collect result
3. Compare Result with expected output model.
4. Collect meaningful results and produce a report.

1. Extends PyUnit = execution engine
2. Provide accurate mismatch reasons

Bottom up testing framework : TUnit

```
class TUnit(unittest.TestCase):

    def should_have(self, model, criteria):
        answer, message = criteria.check(model)
        assert answer ,message

    def should_not_have(self, model, criteria):
        answer,message = criteria.check(model)
        assert not answer, message
```

Many More could be added to this !!!

Bottom up testing framework : TUnit

The example:

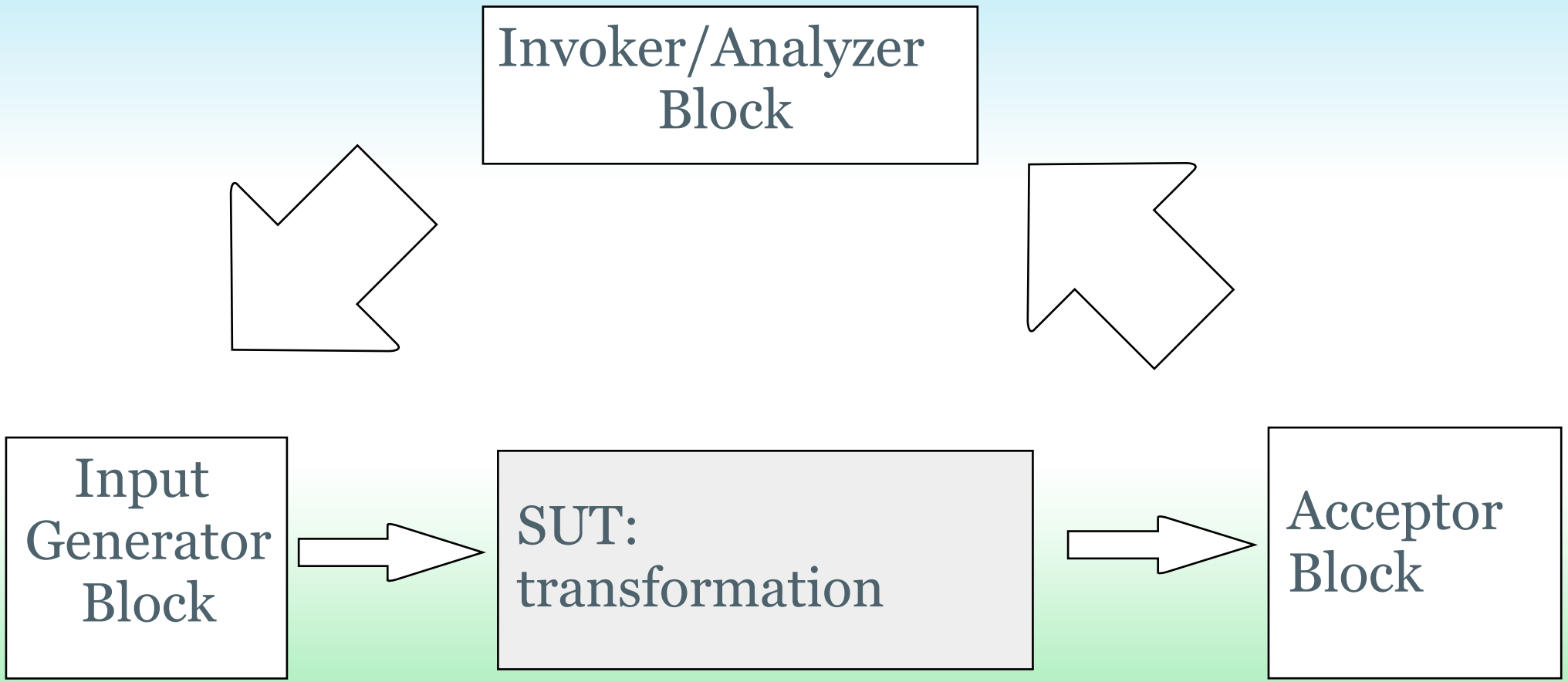
ran the example with 4 criteria .

one criteria failed

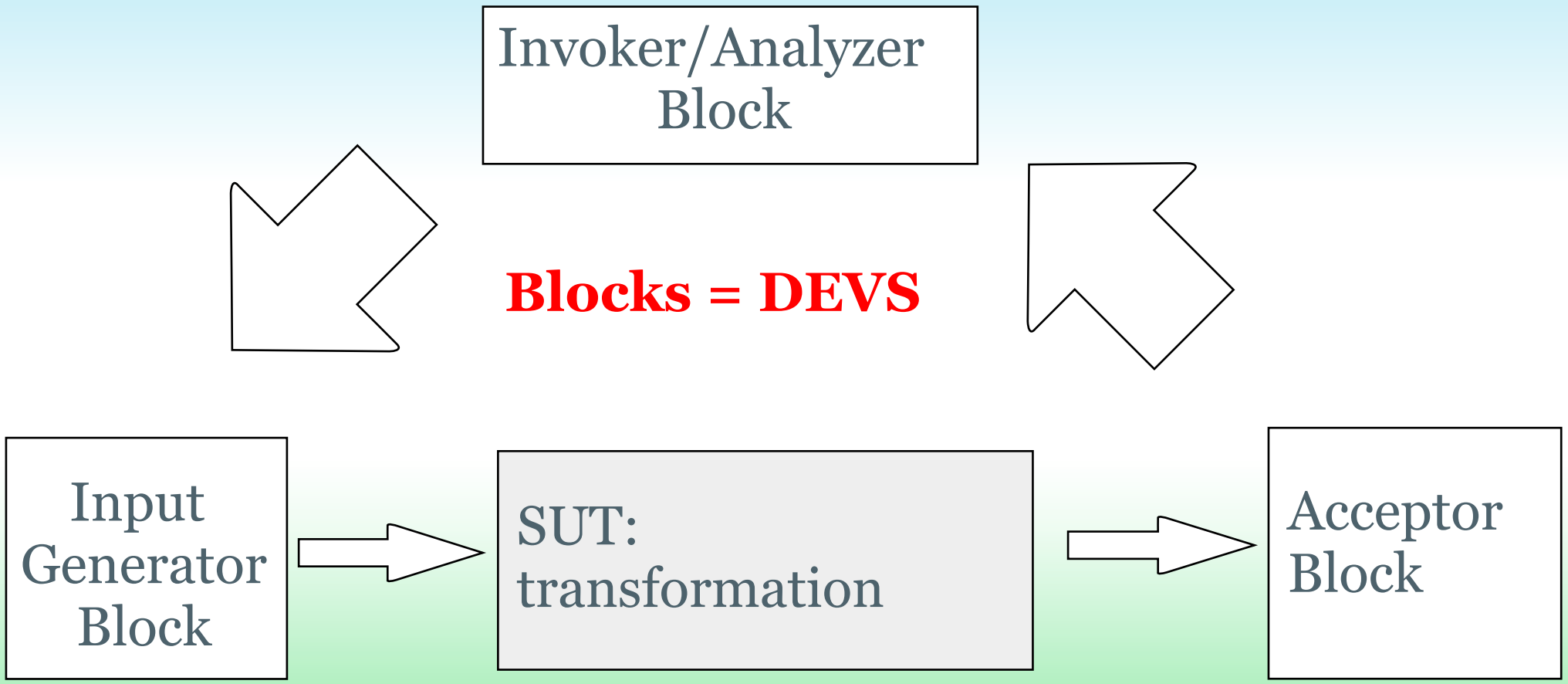
```
<ATOM3.Graph.Graph instance at 0x0206CFA8>
.-------------------- Transforming graph --------------------
<ATOM3.Graph.Graph instance at 0x024580D0>
<ATOM3.Graph.Graph instance at 0x01C33300>
<ATOM3.Graph.Graph instance at 0x02349BE8>
<ATOM3.Graph.Graph instance at 0x02349198>
<ATOM3.Graph.Graph instance at 0x01C55080>
<ATOM3.Graph.Graph instance at 0x02452BC0>
<ATOM3.Graph.Graph instance at 0x021F3670>
<ATOM3.Graph.Graph instance at 0x022F0968>
<ATOM3.Graph.Graph instance at 0x022DE030>
<ATOM3.Graph.Graph instance at 0x01FB2DA0>
<ATOM3.Graph.Graph instance at 0x01E5BBE8>
<ATOM3.Graph.Graph instance at 0x019B8490>
<ATOM3.Graph.Graph instance at 0x02579B70>
<ATOM3.Graph.Graph instance at 0x021E2170>
.
======================================================================
FAIL: testDoesntHaveCapacirtyTwoPlace (__main__.TestingTraffic2PN)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Documents and Settings\Amr\Desktop\my-Devs-Code\DEVS Code\Unit-tests.p
y", line 38, in testDoesntHaveCapacirtyTwoPlace
    self.should_not_have(graph,Pncapacity2segment())
  File "C:\Documents and Settings\Amr\Desktop\my-Devs-Code\DEVS Code\TUnit\TUnit.
py", line 21, in should_not_have
    assert not answer, message
AssertionError: Criteria Pncapacity2segment was matched successfuly.

----------------------------------------------------------------------
Ran 4 tests in 4.078s
```

3. Modeling the model transformation framework: The other direction !?

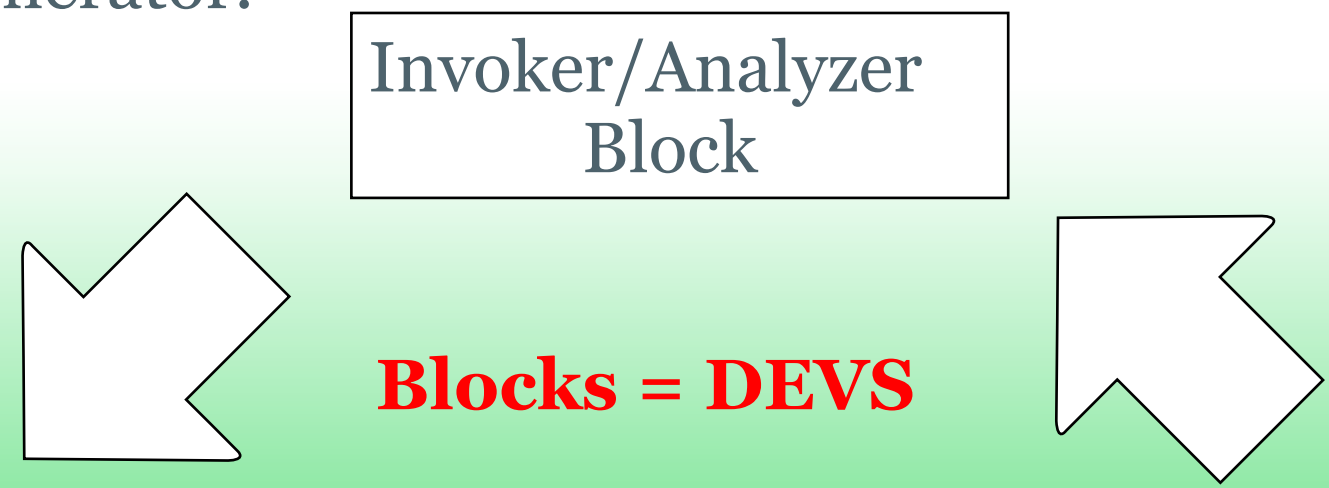


3. Modeling the model transformation framework: The other direction !?



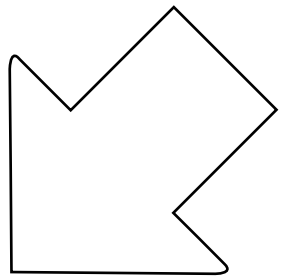
The invoker analyzer Block

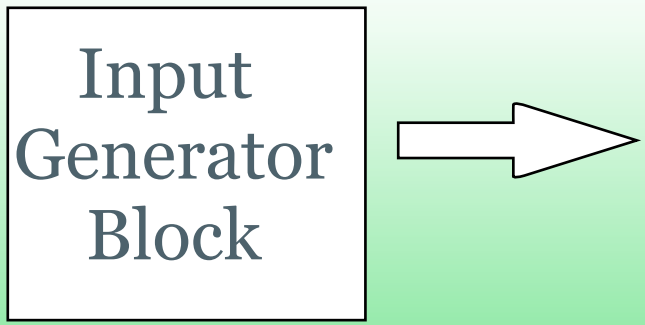
1. Atomic Dev that contain a list of input model files names that need to be tested.
2. After each internal transition it produces the name of the file for the next model to be tested and wait for the Result
3. When it receives the results it
    1. compares the results to the expected results and append to the report .
    2. Reads the next model name and send it to the Model generator.



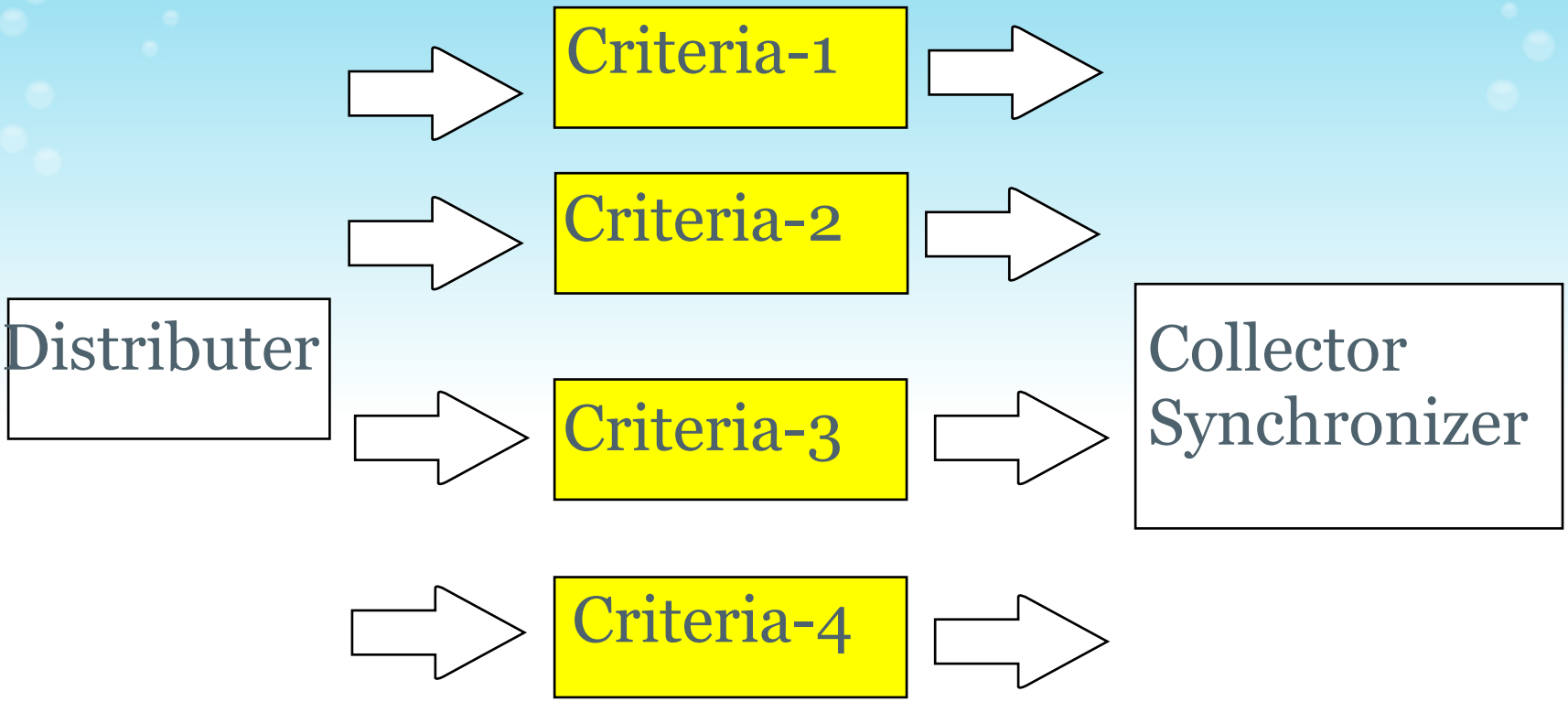
Input Model Generator

1. waits for an event called testCase which has parameters for the file name ,
2. loads the file into a model object (Atom3 maybe)
3. and send it to black box (MoTif Dev) or a black box in a Devs container .



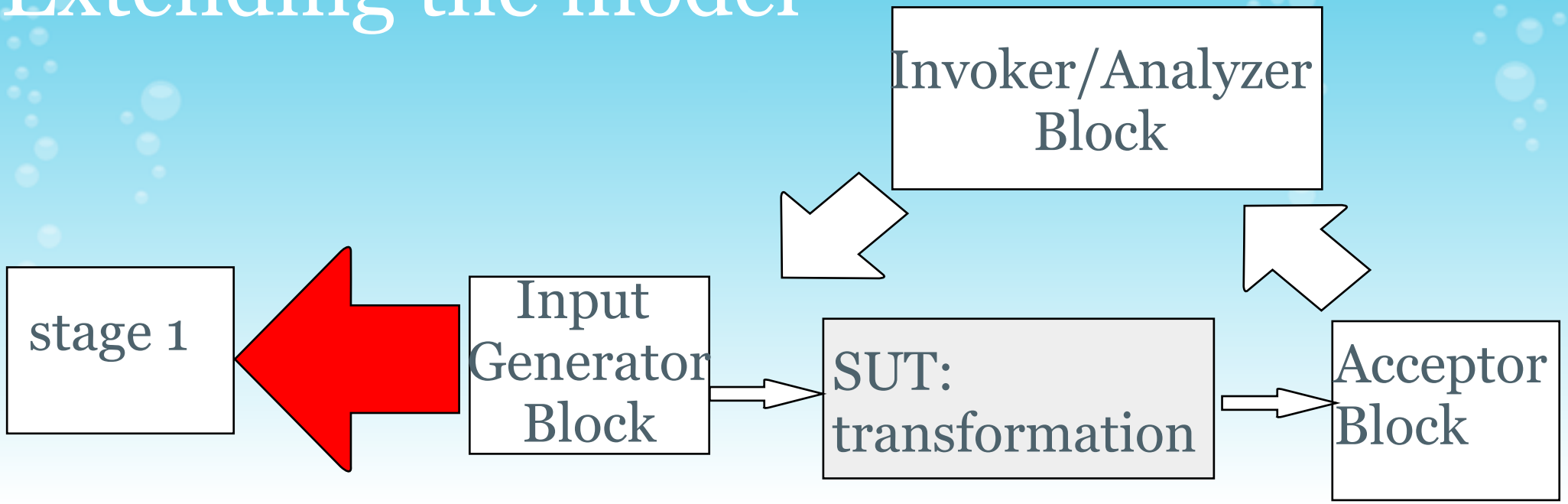


The Acceptor Block dissected



Collector sends the result to Invoker

Extending the model



stage 2

Divide the test exec into multiple stages.

stage 3

Add control structure to you testing.

Testing Model Transformation

Issues/ and comments:

1. Unit Testing approach i.e test cases are already specified.
2. Biggest issue is model comparison. (criteria + XML diff)
3. Integration with Sagar's work .
   3.1 : Need outputs for the generated test cases
   3.2 : Mutation analysis modeling.

Questions ?

Thank you